E 5264

JAPANESE PATENT OFFICE

PATENT ABSTRACTS OF JAPAN

(11)Publication number: 09128346
(43)Date of publication of application: 16.05.1997

(51)Int.Cl.
G06F 15/16
G06F 13/36
G06F 15/163

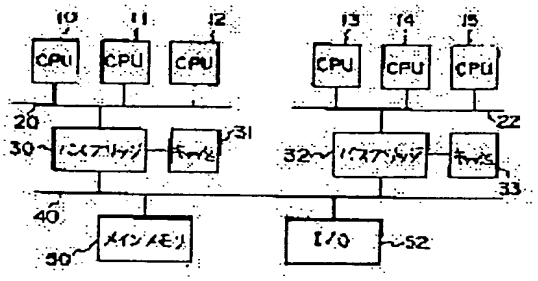
(21)Application number: 07285829
(22)Date of filing: 02.11.1995

(71)Applicant: MITSUBISHI ELECTRIC CORP
(72)Inventor: KAMEMARU TOSHIHISA
YASUNAGA HIROAKI

(54) HIERARCHICAL BUS SYSTEM

(57)Abstract:

PROBLEM TO BE SOLVED: To increase the number of CPU's which can be connected to a system in the multiprocessor system.
SOLUTION: CPU's 10–12 are connected to a bus bridge 30 through a host bus 20 and CPU 13–15 are connected to a bus bridge 32 through a host bus 22. The bus bridges 30 and 32 are connected to a main memory 50 and an I/OF device 52 through a slave bus 40. Bridge caches 31 and 33 shared by host CPU 10–12 and 13–15 are connected to the bus bridges 30 and 32. The host buses 20 and 22 are split buses and CPU's 10–15 can deal with the split. The number of CPU's which can be connected is increased by hierarchization with such constitution, and the deterioration of throughput and latency, accompanying hierarchization, can be prevented by the adoption of the bridge cache and by making the



E5264

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（A）　(11)特許出願公開番号

特開平9－128346

(43)公開日　平成９年(1997)５月16日

(51)Int.Cl.[6]		識別記号	庁内整理番号	ＦＩ		技術表示箇所
Ｇ０６Ｆ	15/16			Ｇ０６Ｆ	15/16	４００Ｂ
	13/36	３１０			13/36	３１０Ｅ
	15/163				15/16	３２０Ｋ

審査請求　未請求　請求項の数42　ＯＬ　（全 48 頁）

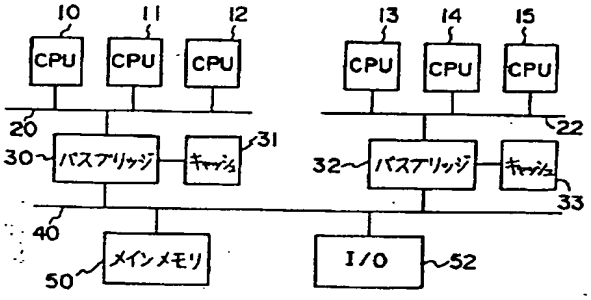
(21)出願番号　特願平7－285829

(22)出願日　平成７年(1995)11月２日

(71)出願人　000006013
三菱電機株式会社
東京都千代田区丸の内二丁目２番３号
(72)発明者　亀丸　敏久
東京都千代田区丸の内二丁目２番３号　三菱電機株式会社内
(72)発明者　安永　裕明
東京都千代田区丸の内二丁目２番３号　三菱電機株式会社内
(74)代理人　弁理士　吉田　研二　（外２名）

(54)【発明の名称】　階層バスシステム

(57)【要約】
【課題】　マルチプロセッサシステムにおいて、システムに接続可能なＣＰＵの数を増加させる。
【解決手段】　ＣＰＵ１０～１２は上位バス２０を介してバスブリッジ３０に接続され、ＣＰＵ１３～１５は上位バス２２を介してバスブリッジ３２に接続される。バスブリッジ３０及び３２とが下位バス４０を介してメインメモリ５０及びＩ／Ｏ装置５２が接続される。バスブリッジ３０及び３２にはそれぞれ上位のＣＰＵ１０～１２及び１３～１５によって共有されるブリッジキャッシュ３１及び３３が接続される。上位バス２０及び２２はスプリットバスであり、ＣＰＵ１０～１５はスプリット対応可能となっている。この構成によれば、階層化により接続可能なＣＰＵ数が増えるとともに、ブリッジキャッシュの採用及び上位バスのスプリットバス化により、階層化に伴うスループットやレイテンシの劣化を防止することができる。



【特許請求の範囲】

【請求項１】　複数のプロセッサとそれらプロセッサによって共有されるメインメモリとを含む階層バスシステムにおいて、
前記プロセッサを複数個ずつ上位バスにて相互に接続し、
各上位バスごとに、当該上位バスに接続されるプロセッサによって共有されるブリッジキャッシュを有するバスブリッジを設け、
各バスブリッジを下位バスにて相互に接続し、この下位バスに前記メインメモリを接続し、
前記プロセッサをスプリット対応可能とし、
前記プロセッサから上位バスに発行されたトランザクションを前記バスブリッジにてスプリット処理することを特徴とする階層バスシステム。

【請求項２】　請求項１記載の階層バスシステムにおいて、
前記プロセッサは、各々プロセッサキャッシュを有し、
前記プロセッサキャッシュ及びブリッジキャッシュは、各キャッシュブロックのステートを記憶するステート記憶手段をそれぞれ有し、
前記プロセッサ及びバスブリッジは、自らが有するプロセッサキャッシュ又はブリッジキャッシュの前記ステート記憶手段を制御するステート制御手段をそれぞれ有し、
前記プロセッサキャッシュの各キャッシュブロックの取り得るステートを、前記ブリッジキャッシュの対応キャッシュブロックのステートに基づいて制約することにより、ブリッジキャッシュとこのブリッジキャッシュの上位のプロセッサキャッシュとのマルチレベル包含性を維持することを特徴とする階層バスシステム。

【請求項３】　請求項２記載の階層バスシステムであって、前記プロセッサキャッシュ及びブリッジキャッシュのステートがＭＥＳＩプロトコルに従う場合において、
ブリッジキャッシュのステートがＩの場合には、上位のプロセッサキャッシュの取り得るステートをＩに限定し、
ブリッジキャッシュのステートがＳの場合には、上位のプロセッサキャッシュの取り得るステートをＳ又はＩに限定し、
ブリッジキャッシュのステートがＥの場合には、上位のプロセッサキャッシュの取り得るステートをＳ又はＩに限定し、
ブリッジキャッシュのステートがＭの場合には、上位のプロセッサキャッシュがすべてのステートを取り得るように認めることを特徴とする階層バスシステム。

【請求項４】　請求項２記載の階層バスシステムであって、前記プロセッサキャッシュ及びブリッジキャッシュのステートがＭＥＳＩプロトコルに従う場合において、
ブリッジキャッシュのステートがＩの場合には、上位のプロセッサキャッシュの取り得るステートをＩに限定し、
ブリッジキャッシュのステートがＳの場合には、上位のプロセッサキャッシュの取り得るステートをＳ又はＩに限定し、
ブリッジキャッシュのステートがＥ又はＭの場合には、上位のプロセッサキャッシュがすべてのステートを取り得るように認めることを特徴とする階層バスシステム。

【請求項５】　請求項３又は請求項４記載の階層バスシステムにおいて、
前記バスブリッジは、上位バスから受け取ったトランザクションの種類と当該トランザクションのアドレスに対応するブリッジキャッシュのステートとに基づき、当該トランザクションをスプリットするか否か及びスプリットする場合において下位バスに発行するトランザクションを決定する下位バス出力判定手段を有し、上位バスからトランザクションを受け取った場合、この下位バス出力判定手段によって決定されたトランザクションを下位バスに発行することを特徴とする階層バスシステム。

【請求項６】　請求項５に記載の階層バスシステムにおいて、
前記下位バス出力判定手段は、
上位バスのトランザクションがリードである場合には、ブリッジキャッシュの対応ブロックのステートがＩである場合にのみ当該トランザクションをスプリットして下位バスにリード・トランザクションを発行し、
上位バスのトランザクションがインバリデートである場合には、ブリッジキャッシュの対応ブロックのステートがＳである場合にのみ当該トランザクションをスプリットして下位バスにインバリデート・トランザクションを発行し、
上位バスのトランザクションがリード・アンド・インバリデートである場合、ブリッジキャッシュの対応ブロックのステートがＳである場合には当該トランザクションをスプリットして下位バスにインバリデート・トランザクションを発行し、ブリッジキャッシュの対応ブロックのステートがＩである場合には当該トランザクションをスプリットして下位バスにリード・アンド・インバリデート・トランザクションを発行するように決定することを特徴とする階層バスシステム。

【請求項７】　請求項６記載の階層バスシステムにおいて、
前記下位バス出力判定手段は、
上位バスのトランザクションがリードである場合に、当該リード・トランザクションが命令コードを要求するものであるかデータを要求するものであるかを判別する要求判別手段を有し、
当該リード・トランザクションがデータを要求するものである場合には下位バスにリード・アンド・インバリデートを発行することを特徴とする階層バスシステム。

【請求項８】　請求項３又は請求項４に記載の階層バスシステムにおいて、
前記バスブリッジは、下位バスに発行したトランザクションの種類と下位バスのスヌープ結果に基づき、ブリッジキャッシュにおける対応キャッシュブロックの次ステートを制御する第１のステート制御手段を有することを特徴とする階層バスシステム。
【請求項９】　請求項８記載の階層バスシステムにおいて、
前記第１のステート制御手段は、
前記トランザクションの種類がインバリデート又はリード・アンド・インバリデートである場合は、前記ブリッジキャッシュの対応ブロックの次ステートをＭにし、
前記トランザクションの種類がリードである場合は、下位バスのスヌープ結果がミスの場合には前記ブリッジキャッシュの対応ブロックの次ステートをＥにし、下位バスのスヌープ結果がＨＩＴ又はＨＩＴＭの場合には前記ブリッジキャッシュの対応ブロックの次ステートをＳにすることを特徴とする階層バスシステム。
【請求項１０】　請求項３記載の階層バスシステムにおいて、
前記バスブリッジは、
自バスブリッジから下位バスに発行したトランザクションの種類に基づきスプリット応答時における上位バスに対するスヌープ出力を生成する上位バススヌープ出力生成手段を有し、
前記上位バススヌープ出力生成手段は、前記トランザクションの種類がインバリデート又はリード・アンド・インバリデートである場合は前記スヌープ出力をミスとし、前記トランザクションの種類がリードである場合は前記スヌープ出力をＨＩＴとし、
バスブリッジが上位バスのトランザクションをスプリットして下位バスにトランザクションを発行した場合に、前記上位バススヌープ出力生成手段からのスヌープ出力によって、上位バスを介して当該バスブリッジに接続されたプロセッサのプロセッサキャッシュのステートを制御することを特徴とする階層バスシステム。
【請求項１１】　請求項４記載の階層バスシステムにおいて、
前記バスブリッジは、
自バスブリッジから下位バスに発行したトランザクションの種類及び当該トランザクションに対する下位バスからのスヌープ結果に基づき、スプリット応答時における上位バスに対するスヌープ出力を生成する上位バススヌープ出力生成手段を有し、
前記上位バススヌープ生成手段は、前記トランザクションの種類がインバリデート又はリード・アンド・インバリデートである場合は前記スヌープ出力をミスとし、前記トランザクションの種類がリードである場合は、下位バスのスヌープ結果がミスの場合は前記スヌープ出力をミスとし下位バスのスヌープ結果がＨＩＴ又はＨＩＴＭの場合は前記スヌープ出力をＨＩＴとし、
バスブリッジが上位バスのトランザクションをスプリットして下位バスにトランザクションを発行した場合に、前記上位バススヌープ出力生成手段からのスヌープ出力によって、上位バスを介して当該バスブリッジに接続されたプロセッサのプロセッサキャッシュのステートを制御することを特徴とする階層バスシステム。
【請求項１２】　請求項３に記載の階層バスシステムにおいて、
前記バスブリッジは、
下位バスから受け取ったトランザクションの種類と当該トランザクションのアドレスに対応するブリッジキャッシュのステートとに基づき、上位バスに発行するトランザクションの種類を決定する上位バス出力判定手段を有し、
前記上位バス出力判定手段は、前記トランザクションの種類がリードである場合はブリッジキャッシュの対応ステートがＭの場合に上位バスにリード・トランザクションを発行すると決定し、前記トランザクションの種類がインバリデート又はリード・アンド・インバリデートである場合は前記ブリッジキャッシュの対応ステートがＭ又はＥ又はＳの場合に上位バスにインバリデート・トランザクションを発行すると決定することを特徴とする階層バスシステム。
【請求項１３】　請求項４に記載の階層バスシステムにおいて、
前記バスブリッジは、
下位バスから受け取ったトランザクションの種類と当該トランザクションのアドレスに対応するブリッジキャッシュのステートとに基づき、上位バスに発行するトランザクションの種類を決定する上位バス出力判定手段を有し、
前記上位バス出力判定手段は、前記トランザクションがリードである場合はブリッジキャッシュの対応ステートがＭ又はＥの場合に上位バスにリード・トランザクションを発行すると決定し、前記トランザクションの種類がインバリデート又はリード・アンド・インバリデートである場合は前記ブリッジキャッシュの対応ステートがＭ又はＥ又はＳの場合に上位バスにインバリデート・トランザクションを発行すると決定することを特徴とする階層バスシステム。
【請求項１４】　請求項１２又は請求項１３記載の階層バスシステムにおいて、
前記バスブリッジは、
ブリッジキャッシュの各キャッシュブロックごとについて、当該キャッシュブロックについての所定のトランザクションを上位バスに対して発行する必要があるか否かの状態を示す状態フラグを記憶した上位キャッシュ状態フラグ記憶手段と、

上位バスに発行されたトランザクションをモニタして、当該トランザクションの種類及びアドレスに基づいて前記上位キャッシュ状態フラグ記憶手段の当該キャッシュブロックの状態を制御するフラグ制御手段と、
下位バスから前記所定トランザクションを受け取った場合に、前記上位キャッシュ状態フラグを参照し、当該所定トランザクションのアドレスに対応するキャッシュブロックの状態フラグが上位バスに対して当該所定トランザクションを発行する必要がないことを示している場合は、当該所定トランザクションの上位バスへの発行を抑制する発行抑制手段と、
を有することを特徴とする階層バスシステム。

【請求項１５】　請求項３記載の階層バスシステムにおいて、
前記バスブリッジは、
前記下位バスから受け取ったトランザクションの種類と、当該トランザクションのアドレスについてのブリッジキャッシュのステートとに基づき、前記下位バスに対するスヌープ出力を生成する下位バススヌープ出力生成手段を有し、
前記下位バススヌープ出力生成手段は、
前記トランザクションの種類がリードの場合において、
前記ブリッジキャッシュの対応ステートがMの場合はＨＩＴＭをスヌープ出力とし、前記ブリッジキャッシュの対応ステートがＥ又はＳの場合はＨＩＴをスヌープ出力とし、
前記トランザクションの種類がインバリデート又はリード・アンド・インバリデートの場合には、前記ブリッジキャッシュの対応ステートがMの場合にＨＩＴＭをスヌープ出力とすることを特徴とする階層バスシステム。

【請求項１６】　請求項４記載の階層バスシステムにおいて、
前記バスブリッジは、
前記下位バスから受け取ったトランザクションの種類と、当該トランザクションのアドレスについてのブリッジキャッシュのステートと、当該トランザクションに対応して上位バスに発行したトランザクションに対する上位バスのスヌープ結果とに基づき、前記下位バスに対するスヌープ出力を生成する下位バススヌープ出力生成手段を有し、
前記下位バススヌープ出力生成手段は、
前記トランザクションの種類がリードの場合において、
前記スヌープ結果がＨＩＴＭの場合は前記ブリッジキャッシュの対応ステートがM又はＥの場合にＨＩＴＭをスヌープ出力とし、前記スヌープ結果がＨＩＴＭ以外の場合は前記ブリッジキャッシュの対応ステートがMの場合にはＨＩＴＭをスヌープ出力とし前記ブリッジキャッシュの対応ステートがＥ又はＳの場合にはＨＩＴをスヌープ出力とし、
前記トランザクションの種類がインバリデート又はリード・アンド・インバリデートの場合において、前記スヌープ結果がＨＩＴＭの場合は前記ブリッジキャッシュの対応ステートがM又はＥの場合にＨＩＴＭをスヌープ出力とし、前記スヌープ結果がＨＩＴＭ以外の場合は前記ブリッジキャッシュの対応ステートがMの場合にＨＩＴＭをスヌープ出力とすることを特徴とする階層バスシステム。

【請求項１７】　請求項３又は請求項４に記載の階層バスシステムにおいて、
前記バスブリッジは、下位バスから受け取ったトランザクションに対応して上位バスに発行したトランザクションの種類に基づき、ブリッジキャッシュの対応ブロックの次ステートを制御する第２のステート制御手段を有し、
前記第２のステート制御手段は、
前記トランザクションの種類がリードである場合は、前記ブリッジキャッシュの対応ブロックの次ステートをＳにし、
前記トランザクションの種類がインバリデート又はリード・アンド・インバリデートである場合は、前記ブリッジキャッシュの対応ブロックの次ステートをＩにすることを特徴とする階層バスシステム。

【請求項１８】　請求項３又は請求項４に記載の階層バスシステムにおいて、
前記バスブリッジは、下位バスから受け取ったトランザクションに対応して上位バスに発行したトランザクションの種類と、当該トランザクションに対する上位バスのスヌープ結果とに基づき、ブリッジキャッシュの対応ブロックの次ステートを制御する第２のステート制御手段を有し、
前記第２のステート制御手段は、
前記トランザクションの種類がリードである場合は、前記上位バスのスヌープ結果がミスの場合には前記ブリッジキャッシュの対応ブロックの次ステートをＩにし、前記上位バスのスヌープ結果がＨＩＴ又はＨＩＴＭの場合には前記ブリッジキャッシュの対応ブロックの次ステートをＳにし、
上位バスに発行したトランザクションがインバリデート又はリード・アンド・インバリデートである場合は、前記ブリッジキャッシュの対応ブロックの次ステートをＩにすることを特徴とする階層バスシステム。

【請求項１９】　請求項３又は請求項４記載の階層バスシステムにおいて、
前記バスブリッジは、
上位バスから受け取ったトランザクションのアドレスに基づき、ブリッジキャッシュにおいて当該アドレスに対応するキャッシュブロックのエビクションが必要か否かを判定するエビクション判定手段と、
エビクションが必要と判定された場合に、前記ブリッジキャッシュのステート記憶手段に記憶されたエビクショ

ン対象のキャッシュブロックのステートに基づき、上位バスに対して前記トランザクションのアドレスに対するインバリデートトランザクションを発行するか否かを判定するインバリデート判定手段であって前記エビクション対象のキャッシュブロックのステートがＭ又はＥ又はＳの場合に上位バスに対してインバリデート・トランザクションを発行すると判定するインバリデート判定手段と、
を有することを特徴とする階層バスシステム。
【請求項２０】　請求項３記載の階層バスシステムにおいて、
前記バスブリッジは、
上位バスから受け取ったトランザクションのアドレスに基づき、ブリッジキャッシュにおいて当該アドレスに対応するキャッシュブロックのエビクションが必要か否かを判定するエビクション判定手段と、
エビクションが必要と判定された場合に、前記ステート記憶手段に記憶されたエビクション対象のキャッシュブロックのステートに基づき下位バスに前記トランザクションのアドレスに対するライトバックトランザクションを発行するか否かを判定するライトバック判定手段であって、エビクション対象のキャッシュブロックのステートがＭの場合のみ下位バスにライトバックトランザクションを発行すると判定するライトバック判定手段と、
を有することを特徴とする階層バスシステム。
【請求項２１】　請求項４記載の階層バスシステムにおいて、
前記バスブリッジは、
上位バスから受け取ったトランザクションのアドレスに基づき、ブリッジキャッシュにおいて当該アドレスに対応するキャッシュブロックのエビクションが必要か否かを判定するエビクション判定手段と、
エビクションが必要と判定された場合に、上位バスに対して前記キャッシュブロックに対するインバリデートトランザクションを発行するインバリデート発行手段と、
エビクションが必要と判定された場合に、前記ステート記憶手段に記憶されたエビクション対象のキャッシュブロックのステートと前記インバリデート発行手段から発行されたインバリデートトランザクションに対する上位バスのスヌープ結果とに基づき下位バスに前記エビクション対象のキャッシュブロックに対するライトバックトランザクションを発行するか否かを判定するライトバック判定手段と、
を有し、
前記ライトバック判定手段は、上位バスのスヌープ結果がＨＩＴＭの場合はエビクション対象のキャッシュブロックのステートがＭ又はＥのときにライトバックを発行すると判定し、上位バスのスヌープ結果がＨＩＴＭ以外の場合はエビクション対象のキャッシュブロックのステートがＭのときにライトバックを発行すると判定することを特徴とする階層バスシステム。
【請求項２２】　請求項１９～請求項２１のいずれかに記載の階層バスシステムにおいて、
前記バスブリッジは、
ブリッジキャッシュの前記エビクション対象のキャッシュブロックのデータを格納するバッファ手段を有し、
エビクション判定手段にてエビクションが必要と判定されると、ブリッジキャッシュのエビクション対象のキャッシュブロックのデータを前記バッファ手段に退避させたのちブリッジキャッシュの当該キャッシュブロックを用いて前記上位バスからのトランザクションに対する処理を実行し、前記バッファ手段に退避したデータを用いてライトバック処理を行うことを特徴とする階層バスシステム。
【請求項２３】　請求項２２記載の階層バスシステムにおいて、
前記バッファ手段は、
エビクション対象のキャッシュブロックのアドレスを格納するアドレスレジスタと、
エビクション対象のキャッシュブロックのデータを格納するデータバッファと、
を有することを特徴とする階層バスシステム。
【請求項２４】　請求項２２記載の階層バスシステムにおいて、
前記バッファ手段は、
トランザクションのアドレス指定によってアクセス可能な補助キャッシュとして構成されることを特徴とする階層バスシステム。
【請求項２５】　請求項２２記載の階層バスシステムにおいて、
前記バスブリッジは、
エビクション判定に応じて当該バスブリッジから上位バスに発行したインバリデートトランザクションに対して、上位バスのプロセッサから該当キャッシュブロックのデータがライトバックされてきた場合に、プロセッサからライトバックされてきたデータによって前記バッファ手段に格納されているデータを更新することを特徴とする階層バスシステム。
【請求項２６】　請求項１記載の階層バスシステムにおいて、
前記バスブリッジは、
当該バスブリッジにおいてスプリット中のトランザクションの有無を示すスプリットフラグ手段と、
上位バスから受け取ったトランザクションの種類が、キャッシュブロックを使用する種類であるか否かを判定するトランザクション判定手段と、
上位バスから受け取ったトランザクションのリトライの要否を判定するリトライ判定手段であって、前記スプリットフラグ手段がスプリット中のトランザクションが有ることを示し、かつ上位バスから受け取ったトランザク

ションが前記トランザクション判定手段によってキャッシュブロックを使用する種類であると判定された場合にリトライが必要と判定するリトライ判定手段と、
前記リトライ判定手段によってリトライが必要と判定されたときに、上位バスに対して前記トランザクションについてのリトライ信号を出力するリトライ出力手段と、
を有し、
前記上位バスのトランザクションを発行したプロセッサは、バスブリッジからリトライ信号を受け取った場合に、当該トランザクションをいったん終了し、所定時間後に前記トランザクションを再発行することを特徴とする階層バスシステム。
【請求項２７】　請求項１記載の階層バスシステムにおいて、
前記バスブリッジは、
当該バスブリッジにおいてスプリット中のトランザクションが使用するキャッシュブロックのアドレス情報を記憶するスプリット情報記憶手段と、
上位バスから受け取ったトランザクションのアドレスと前記スプリット情報記憶手段に記憶されたアドレス情報とを比較し、上位バスから受け取ったトランザクションがスプリット中のトランザクションの使用するキャッシュブロックを使用するか否かを判定するアドレス判定手段と、
上位バスから受け取ったトランザクションのリトライの要否を判定するリトライ判定手段であって、前記アドレス判定手段において上位バスから受け取ったトランザクションがスプリット中のトランザクションの使用するキャッシュブロックを使用すると判定された場合に、リトライが必要と判定するリトライ判定手段と、
を有し、
前記上位バスのトランザクションを発行したプロセッサは、バスブリッジからリトライ信号を受け取った場合に、当該トランザクションをいったん終了し、所定時間後に前記トランザクションを再発行することを特徴とする階層バスシステム。
【請求項２８】　請求項２１に記載の階層バスシステムにおいて、
前記バスブリッジは、
上位バスから受け取ったトランザクションのアドレスとバッファ手段に退避中のキャッシュブロックのアドレスとを比較し、当該トランザクションが退避中のキャッシュブロックに対するトランザクションであるか否かを判定するアドレス比較手段と、
前記上位バスからのトランザクションが退避中のキャッシュブロックに対するトランザクションだと判定された場合に、当該トランザクションに対してリトライが必要と判定するリトライ判定手段と、
前記リトライ判定手段によってリトライが必要と判定されたときに、上位バスに対して前記トランザクションについてのリトライ信号を出力するリトライ出力手段と、
を有し、
前記上位バスのトランザクションを発行したプロセッサは、バスブリッジからリトライ信号を受け取った場合に、当該トランザクションをいったん終了し、所定時間後に前記トランザクションを再発行することを特徴とする階層バスシステム。
【請求項２９】　請求項１記載の階層バスシステムにおいて、
前記バスブリッジは、
上位バスから受け取ったトランザクションをスプリットして下位バスにトランザクションを発行した場合において、当該下位バスのトランザクションに対して下位バスからリトライ信号を受け取った場合には、前記上位バスに対してスプリット応答トランザクションを発行すると共に前記上位バスのトランザクション発行元のプロセッサに対してリトライ信号を出力し、
前記トランザクション発行元のプロセッサは、前記バスブリッジからのリトライ信号を受信すると、前記トランザクションをいったん終了し、所定時間後にリトライすることを特徴とする階層バスシステム。
【請求項３０】　請求項１記載の階層バスシステムにおいて、
前記バスブリッジは、
上位バストランザクションが有するスプリット識別子に対して下位バスにおいて当該バスブリッジに固有のブリッジ識別子を付加することにより、下位バストランザクションのスプリット識別子を生成するスプリット識別子拡張手段を有し、
上位バスから受け取った上位バストランザクションをスプリットして下位バスに下位バストランザクションを発行する場合に、前記スプリット識別子拡張手段によって得られたスプリット識別子を下位バストランザクションに付することを特徴とする階層バスシステム。
【請求項３１】　請求項１記載の階層バスシステムにおいて、
前記バスブリッジは、
一方のバスから受け取ったトランザクションのスプリット識別子を所定ビット数のブリッジトランザクション識別子に変換し、他方のバスにおいて当該バスブリッジに固有なブリッジ識別子に前記ブリッジトランザクション識別子を付加することにより、他方のバスに発行するトランザクションに付与するスプリット識別子を生成するスプリット識別子生成手段を有し、
前記ブリッジトランザクション識別子のビット数と前記ブリッジ識別子のビット数との和は前記一方のバスにおけるスプリット識別子のビット数に等しく、
一方のバスから受け取ったトランザクションに対し他方のバスにトランザクションを発行する必要がある場合には、当該他方のバスに発行するトランザクションに対し

て前記スプリット識別子生成手段によって得られたスプリット識別子を当該他方のバスに発行するトランザクションに付することを特徴とする階層バスシステム。

【請求項３２】　請求項３１記載の階層バスシステムにおいて、

前記プロセッサは、上位バスにトランザクションを発行する場合に、当該プロセッサが接続された上位バスにおいて当該プロセッサに固有なプロセッサ識別子に対して、当該プロセッサ内において当該トランザクションに固有なプロセッサトランザクション識別子を付加することにより、当該トランザクションのスプリット識別子を生成し、

前記ブリッジ識別子を前記プロセッサ識別子と同ビット数としたことを特徴とする階層バスシステム。

【請求項３３】　請求項３２記載の階層バスシステムにおいて、

前記バスブリッジのスプリット識別子生成手段は、

当該バスブリッジが下位バスに発行するトランザクションに対して付与可能なブリッジトランザクション識別子の使用・未使用状態を管理する下位バス用管理テーブルを有し、

上位バスからのトランザクションに対して下位バスにトランザクションの発行が必要な場合は、前記下位バス用管理テーブルにおいて未使用のブリッジトランザクション識別子を求め、求められたブリッジトランザクション識別子を用いて前記下位バスに発行するトランザクションに対するスプリット識別子を生成するとともに、前記上位バスからのトランザクションのスプリット識別子を当該ブリッジトランザクション識別子に対応づけて前記下位バス用管理テーブルに登録することを特徴とする階層バスシステム。

【請求項３４】　請求項３３記載の階層バスシステムにおいて、

前記バスブリッジは、上位バスから受け取ったトランザクションに対して下位バスにトランザクションの発行が必要な場合において、前記下位バス用管理テーブルに未使用のブリッジトランザクション識別子がない場合は、上位バスに対してリトライ終了信号を発行し、当該上位バスのトランザクションの発行元プロセッサに対して当該トランザクションのリトライ処理を行わせることを特徴とする階層バスシステム。

【請求項３５】　請求項３２記載の階層バスシステムにおいて、

前記バスブリッジのスプリット識別子生成手段は、

当該バスブリッジが上位バスに発行するトランザクションに対して付与可能なブリッジトランザクション識別子の使用・未使用状態を管理する上位バス用管理テーブルを有し、

下位バスからのトランザクションに対して上位バスにトランザクションの発行が必要な場合は、前記上位バス用管理テーブルにおいて未使用のブリッジトランザクション識別子を求め、求められたブリッジトランザクション識別子を用いて前記上位バスに発行するトランザクションに対するスプリット識別子を生成するとともに、前記下位バスからのトランザクションのスプリット識別子を当該ブリッジトランザクション識別子に対応づけて前記上位バス用管理テーブルに登録することを特徴とする階層バスシステム。

【請求項３６】　請求項３５記載の階層バスシステムにおいて、

前記バスブリッジは、下位バスから受け取ったトランザクションに対して上位バスにトランザクションの発行が必要な場合において、前記上位バス用管理テーブルに未使用のブリッジトランザクション識別子がない場合は、前記上位バス用管理テーブルに未使用のブリッジトランザクション識別子ができるのを待って上位バスに対するトランザクションの発行を行うことを特徴とする階層バスシステム。

【請求項３７】　請求項１記載の階層バスシステムにおいて、

バスの物理的仕様及び論理的仕様を前記下位バスと上位バスとで共通としたことを特徴とする階層バスシステム。

【請求項３８】　請求項３７記載の階層バスシステムにおいて、

前記下位バスは、１個以上のボードインタフェースを有し、

これら各ボードインタフェースに対して、１つのプロセッサを実装した第１のボードと、複数のプロセッサ及び上位バス及びバスブリッジを実装した第２のボードと、のいずれかを接続することによりシステム構成を変更可能としたことを特徴とする階層バスシステム。

【請求項３９】　請求項１記載の階層バスシステムにおいて、

前記下位バスを複数設け、これら各下位バスにそれぞれメインメモリを接続し、前記各上位バスに対して下位バスと同数のバスブリッジを接続し、前記各上位バスと各下位バスとを互いに１つのバスブリッジを介して接続したことを特徴とする階層バスシステム。

【請求項４０】　請求項３９記載の階層バスシステムにおいて、

前記複数の下位バスのそれぞれに対してバスブリッジを設け、これらバスブリッジを介して前記各下位バスを１つのＩ／Ｏバスに接続し、このＩ／ＯバスにＩ／Ｏ装置を接続したことを特徴とする階層バスシステム。

【請求項４１】　請求項３９又は請求項４０に記載の階層バスシステムにおいて、

同一の上位バスに接続された各バスブリッジは、それぞれ受け持ちのアドレス範囲が定められており、上位バスから受け取ったトランザクションのアドレスが自らの受

け持ちアドレス範囲に含まれる場合にのみ、当該トランザクションに対する処理を行うことを特徴とする階層バスシステム。

【請求項４２】　請求項４１記載の階層バスシステムにおいて、

同一の上位バスに接続された各バスブリッジのアドレス受け持ち範囲を変更可能としたことを特徴とする階層バスシステム。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、複数のプロセッサ及びメモリを含むマルチプロセッサシステムにおける階層的なバスシステムに関する。

【０００２】

【従来の技術】情報処理システムの処理能力に対する要求の増大に伴い、１つのプロセッサでは能力的に不足する場合が出てきている。このような問題に対処するため、複数のプロセッサを連結して並列動作させるマルチプロセッサ方式のシステムが研究・開発されている。

【０００３】マルチプロセッサシステムにおけるプロセッサの結合方式としては従来より様々な方式が考案されているが、実現の容易さや実行するプログラムのプログラミングの容易さなどの理由から、近年ではバス共有型の密結合マルチプロセッサ方式が多く採用されている。図２０は、バス共有型の密結合マルチプロセッサシステムを示す概略図である。図２０のシステムでは、複数のＣＰＵ１０～１２を、バス９０を介してメインメモリ５０やＩ／Ｏ装置５２に共通接続している。

【０００４】

【発明が解決しようとする課題】マルチプロセッサシステムにおいて性能の向上を図る場合、プロセッサ（ＣＰＵ）の並列数を増やすという方法が考えられるが、バス共有型の場合、１本のバスに接続し得るＣＰＵの数には物理的な制限があった。すなわち、１本のバス上に接続するＣＰＵの数を増やすと信号線の電気的な容量負荷の増大や信号線の長さの増加によって反射等の問題が発生するので、バス共有型でしかも各ＣＰＵを高速動作させようとした場合、並列可能なＣＰＵの数には制限があった。また、ＣＰＵ数の増加は、いわゆるバスコンテンションの問題を引き起こすため、この点からもＣＰＵ数は制限されていた。

【０００５】本発明は、このような問題を解決するためになされたものであり、多数のＣＰＵを並列可能な階層バスシステムを提供することを目的とする。

【０００６】

【課題を解決するための手段】前述の目的を達成するために、本発明に係る階層バスシステムは、複数のプロセッサとそれらプロセッサによって共有されるメインメモリとを含む階層バスシステムにおいて、前記プロセッサを複数個ずつ上位バスにて相互に接続し、各上位バスごとに、当該上位バスに接続されるプロセッサによって共有されるブリッジキャッシュを有するバスブリッジを設け、各バスブリッジを下位バスにて相互に接続し、この下位バスに前記メインメモリを接続し、前記プロセッサをスプリット対応可能とし、前記プロセッサから上位バスに発行されたトランザクションを前記バスブリッジにてスプリット処理することを特徴とする。

【０００７】この構成では、まず第１に、プロセッサとメインメモリとの間が上位バス及び下位バスからなる階層的なバス構成となっているため、各上位バスに接続されるプロセッサの数が限られる場合でも、上位バスの数を増やすことによりシステム全体として多数のプロセッサを接続することができる。

【０００８】第２に、この構成では、各バスブリッジに対して当該バスブリッジの上位バスに接続されるプロセッサによって共有されるブリッジキャッシュを設けたことにより、プロセッサから発行されたトランザクションの要求データがブリッジキャッシュに存在する場合には、下位バスを介してメインメモリを読みに行く必要がなくなるため、下位バスのバス負荷を低減し、下位バスのスループットを向上させることができるとともに、プロセッサがトランザクションに対するレスポンスを得るまでの遅れ時間（レイテンシ）を短くすることができる。

【０００９】第３に、この構成では、上位バスをスプリットバスとしたことにより、上位バスのスループットを向上させることができる。ここでスプリットとは、トランザクション発行のサイクルとそのトランザクションに対する応答のサイクルとを分割可能とし、その間にトランザクション発行元のプロセッサにバス使用権を放棄させ、他のバスマスタがトランザクションを発行できるようにする方式である。

【００１０】プロセッサのトランザクションの要求データがブリッジキャッシュに存在しない場合、下位バスにトランザクションを転送してメインメモリや他のクラスタのキャッシュを読みに行く必要がある。この場合において、下位バスからの応答を待つ間、そのトランザクションを発したプロセッサが上位バスを占有していると、上位バスのスループットが低下してしまう。これに対して、本発明の構成では、プロセッサのトランザクションの要求データがブリッジキャッシュに存在せず下位バスにトランザクションを転送する必要がある場合には、そのトランザクションをスプリットすることができるので、その間上位バスに他のトランザクションを発行することができ、全体として上位バスのスループットを向上させることができる。

【００１１】また、本発明は、さらに上位バス上のプロセッサが、それぞれ自分専用のプロセッサキャッシュを有する場合において、プロセッサキャッシュ及びブリッジキャッシュは各キャッシュブロックのステートを記憶

するステート記憶手段をそれぞれ有し、プロセッサ及びバスブリッジは自らが有するプロセッサキャッシュ又はブリッジキャッシュのステート記憶手段を制御するステート制御手段をそれぞれ有し、プロセッサキャッシュの各キャッシュブロックの取り得るステートを、ブリッジキャッシュの対応キャッシュブロックのステートに基づいて制約することにより、ブリッジキャッシュとこのブリッジキャッシュの上位のプロセッサキャッシュとのマルチレベル包含性を維持することを特徴とする。

【００１２】この構成では、プロセッサキャッシュ及びブリッジキャッシュのそれぞれに各キャッシュブロックのステートを記憶するステート記憶手段を設け、このステート記憶手段に記憶されたステートを用いてトランザクションの制御を行う。ステート記憶手段における各ステートは、ステート制御手段によって制御する。ここで、この構成では、プロセッサキャッシュの各キャッシュブロックの取り得るステートをブリッジキャッシュの対応キャッシュブロックのステートに基づいて制約することにより、ブリッジキャッシュとこのブリッジキャッシュの上位のプロセッサキャッシュとのマルチレベル包含性を維持する。この結果、プロセッサキャッシュとブリッジキャッシュとメインメモリとのキャッシュコンシステンシを保つことができる。しかも、プロセッサキャッシュとブリッジキャッシュとの間でマルチレベル包含性を維持することにより、キャッシュコンシステンシ維持のためのバスブリッジの構成を比較的簡単なものとすることができる。

【００１３】また、本発明は、バスブリッジに対してブリッジキャッシュのエビクション対象のキャッシュブロックのデータを格納するバッファ手段を設け、エビクション判定手段にてエビクションが必要と判定されると、ブリッジキャッシュのエビクション対象のキャッシュブロックのデータを前記バッファ手段に退避させたのちブリッジキャッシュの当該キャッシュブロックを用いて上位バスからのトランザクションに対する処理を実行し、前記バッファ手段に退避したデータを用いてライトバック処理を行うことを特徴とする。

【００１４】この構成では、まずバスブリッジが上位バスからトランザクションを受け取った場合、エビクション判定手段によって当該トランザクションのアドレスに対応するブリッジキャッシュのキャッシュブロックをエビクションする必要があるか否かを判定する。この判定の結果エビクションが必要と判定された場合、バスブリッジは、ブリッジキャッシュの当該エビクション対象のキャッシュブロックをバッファ手段に書き込む。この結果、当該エビクション対象のキャッシュブロックのデータのライトバック処理はバッファ手段に書き込まれたデータによって行うことができるので、ブリッジキャッシュの当該エビクション対象のキャッシュブロックのデータを確保しておく必要がなくなる。したがって、当該キャッシュブロックは上位バスからのトランザクションによって使用可能な状態となるので、バスブリッジは、ライトバック処理の終了を待たずに当該キャッシュブロックを用いて上位バストランザクションに対する処理を進行することができる。

【００１５】このように、本構成によれば、トランザクションに対する処理をエビクションに先行して行うことができるため、プロセッサが最初にトランザクションを発行してからそのトランザクションが完結するまでの時間を大幅に短縮することができる。

【００１６】なお、本構成において、さらにエビクション判定に応じてバスブリッジから上位バスに発行したインバリデートトランザクションに対して、上位バスのプロセッサから当該エビクション対象のキャッシュブロックのデータがライトバックされてきた場合に、プロセッサからライトバックされてきたデータによって前記バッファ手段に格納されているデータを更新する構成とすることもできる。この構成によれば、当該バスブリッジの上位のプロセッサに当該エビクション対象のキャッシュブロックの最新のデータが存在する場合に、この最新のデータはバスブリッジのバッファ手段に書き戻されるので、バスブリッジにおけるライトバックにおいては、その最新データをバッファ手段からメインメモリに書き戻すことができる。

【００１７】また、本発明は、バスブリッジが、当該バスブリッジにおいてスプリット中のトランザクションの有無を示すスプリットフラグ手段と、上位バスから受け取ったトランザクションの種類がキャッシュブロックを使用する種類であるか否かを判定するトランザクション判定手段と、上位バスから受け取ったトランザクションのリトライの要否を判定するリトライ判定手段であって、前記スプリットフラグ手段がスプリット中のトランザクションが有ることを示し、かつ上位バスから受け取ったトランザクションが前記トランザクション判定手段によってキャッシュブロックを使用する種類であると判定された場合にリトライが必要と判定するリトライ判定手段と、前記リトライ判定手段によってリトライが必要と判定されたときに、上位バスに対して前記トランザクションについてのリトライ信号を出力するリトライ出力手段とを有し、前記上位バスのトランザクションを発行したプロセッサは、バスブリッジからリトライ信号を受け取った場合に、当該トランザクションをいったん終了し、所定時間後に前記トランザクションを再発行することを特徴とする。

【００１８】この構成において、スプリット中とは、バスブリッジが上位バスからのトランザクションをスプリットしてから、当該トランザクションの発行元のプロセッサに対してスプリット応答が完了するまでの間のことをいう。この構成では、バスブリッジがあるトランザクションについてスプリット中の間に別のトランザクショ

ンを受け取った場合、当該別のトランザクションがキャッシュブロックを使用するかしないかを判定し、使用すると判定される場合にのみ当該別のトランザクションをリトライさせる。そして、キャッシュブロックを使用しない種類のトランザクションについては、リトライさせずにそのまま処理を行う。したがって、バスブリッジに起因するリトライの発生頻度の増大を防止し、システム全体としての処理効率を高めることができる。

【００１９】また、本発明は、バスブリッジが、当該バスブリッジにおいてスプリット中のトランザクションが使用するキャッシュブロックのアドレス情報を記憶するスプリット情報記憶手段と、上位バスから受け取ったトランザクションのアドレスと前記スプリット情報記憶手段に記憶されたアドレス情報とを比較し、上位バスから受け取ったトランザクションがスプリット中のトランザクションの使用するキャッシュブロックを使用するか否かを判定するアドレス判定手段と、上位バスから受け取ったトランザクションのリトライの要否を判定するリトライ判定手段であって、前記アドレス判定手段において上位バスから受け取ったトランザクションがスプリット中のトランザクションの使用するキャッシュブロックを使用すると判定された場合にリトライが必要と判定するリトライ判定手段とを有し、前記上位バスのトランザクションを発行したプロセッサは、バスブリッジからリトライ信号を受け取った場合に、当該トランザクションをいったん終了し、所定時間後に前記トランザクションを再発行することを特徴とする。

【００２０】この構成では、スプリット情報記憶手段に格納されたアドレス情報を用いることにより、バスブリッジが受け取った上位バスからのトランザクションが現在スプリット中のトランザクションの使用するキャッシュブロックを使用するものか否かを判定することができる。そして、この構成では、上位バスからのトランザクションが、現在スプリット中のトランザクションの使用するキャッシュブロックを使用すると判定された場合にのみ、当該上位バスからのトランザクションのリトライを行う。したがって、この構成によれば、バスブリッジに起因するリトライの発生頻度が小さくなり、システム全体としての処理効率を高めることができる。

【００２１】また、本発明は、前記バッファ手段を有する構成において、さらにバスブリッジが、上位バスから受け取ったトランザクションのアドレスとバッファ手段に退避中のキャッシュブロックのアドレスとを比較し、当該トランザクションが退避中のキャッシュブロックに対するトランザクションであるか否かを判定するアドレス比較手段と、前記上位バスからのトランザクションが退避中のキャッシュブロックに対するトランザクションだと判定された場合に、当該トランザクションに対してリトライが必要と判定するリトライ判定手段と、前記リトライ判定手段によってリトライが必要と判定されたときに、上位バスに対して前記トランザクションについてのリトライ信号を出力するリトライ出力手段とを有し、前記上位バスのトランザクションを発行したプロセッサは、バスブリッジからリトライ信号を受け取った場合に、当該トランザクションをいったん終了し、所定時間後に前記トランザクションを再発行することを特徴とする。

【００２２】この構成は、エビクション対象のキャッシュブロックをブリッジキャッシュから一時的に退避しておくためのバッファ手段を有するバスブリッジを用いるシステムに関するものである。バッファ手段に退避しているデータは、データの整合性を保つため他のトランザクションからは読み書きができない。したがって、バスブリッジが当該退避データに対する別のトランザクションを受け取った場合、当該トランザクションについてリトライを行う必要がある。そこで、本構成では、アドレス比較手段によって、上位バスから受け取ったトランザクションのアドレスとバッファ手段に退避中のデータのアドレスとを比較し、退避中のデータに対するトランザクションのみをリトライさせる。この構成によれば、バッファ手段に退避中のデータに対するトランザクションのみをリトライさせることができる。

【００２３】また、本発明は、バスブリッジが、上位バスから受け取ったトランザクションをスプリットして下位バスにトランザクションを発行した場合において、当該下位バスのトランザクションに対して下位バスからリトライ信号を受け取った場合には、前記上位バスに対してスプリット応答トランザクションを発行すると共に前記上位バスのトランザクション発行元のプロセッサに対してリトライ信号を出力し、前記トランザクション発行元のプロセッサは、前記バスブリッジからのリトライ信号を受信すると、前記トランザクションをいったん終了し、所定時間後にリトライすることを特徴とする。

【００２４】この構成は、バスブリッジが、上位バスから受け取ったトランザクションに対応して下位バスにトランザクションを発行した場合において、当該下位バス・トランザクションに対して下位バスのエージェント（例えばＩ／Ｏ装置）などからリトライ信号がバスブリッジに返ってきた場合の処理に関する。すなわち、この構成では、バスブリッジ自身が当該下位バス・トランザクションについてのリトライ処理を行うのではなく、バスブリッジからトランザクション発行元のプロセッサに対してリトライ信号を発行し、そのプロセッサにリトライ処理をさせる。この構成によれば、バスブリッジがリトライ処理のための特別の構成を持つ必要がなくなり、バスブリッジの構成を簡単化することができる。

【００２５】また、本発明は、バスブリッジが、上位バストランザクションが有するスプリット識別子に対して下位バスにおいて当該バスブリッジに固有のブリッジ識別子を付加することにより、下位バストランザクション

のスプリット識別子を生成するスプリット識別子拡張手段を有し、上位バスから受け取った上位バストランザクションをスプリットして下位バスに下位バストランザクションを発行する場合に、前記スプリット識別子拡張手段によって得られたスプリット識別子を下位バストランザクションに付することを特徴とする。

【００２６】この構成において、上位バストランザクションのスプリット識別子は、トランザクションの発行元のプロセッサを識別するための識別子と当該プロセッサ内においてトランザクションを識別するための識別子とを含んでいるが、スプリット識別子のビット数が定まっていれば識別可能なプロセッサの数、すなわちシステムに接続可能なプロセッサの数も自ずと制限される。この接続可能プロセッサ数の論理的制限を緩和するのが本構成である。

【００２７】この構成では、バスブリッジにて上位バストランザクションをスプリットして下位バスにトランザクションを発行する場合には、上位バストランザクションに与えられたスプリット識別子に対してバスブリッジのブリッジ識別子を付加することにより拡張したスプリット識別子を下位バストランザクションに与える。この構成によれば、下位バストランザクションは、拡張されたスプリット識別子に含まれるブリッジ識別子よりその発行元のバスブリッジを特定することが可能となる。したがって、下位バストランザクションからバスブリッジが特定されれば、トランザクション発行元プロセッサはそのバスブリッジの上位にあることになるので、別々の上位バス上のプロセッサに同一のプロセッサ識別子を与えたとしてもそれらを互いに識別することが可能となる。したがって、この構成によれば、プロセッサ識別子の数に制限があっても、ブリッジ識別子を加えたことにより、システム全体として識別可能なプロセッサの数を増やすことができ、階層バスシステム全体に接続可能なプロセッサ数を増やすことができる。

【００２８】また、本発明は、バスブリッジが、一方のバスから受け取ったトランザクションのスプリット識別子を所定ビット数のブリッジトランザクション識別子に変換し、他方のバスにおいて当該バスブリッジに固有なブリッジ識別子に前記ブリッジトランザクション識別子を付加することにより、他方のバスに発行するトランザクションに付与するスプリット識別子を生成するスプリット識別子生成手段を有し、前記ブリッジトランザクション識別子のビット数と前記ブリッジ識別子のビット数との和は前記一方のバスにおけるスプリット識別子のビット数に等しく、一方のバスから受け取ったトランザクションに対し他方のバスにトランザクションを発行する必要がある場合には、当該他方のバスに発行するトランザクションに対して前記スプリット識別子生成手段によって得られたスプリット識別子を当該他方のバスに発行するトランザクションに付することを特徴とする。

【００２９】この構成では、バスブリッジが一方のバスから受け取ったトランザクションも対応して他方のバスにトランザクションを発行する際に当該他方のバスのトランザクションに与えるスプリット識別子を、前記一方のバスのトランザクションのスプリット識別子と同ビット数とする。ここで、バスブリッジがトランザクションに対して与えるスプリット識別子には、当該バスブリッジのブリッジ識別子を含める。このため、バスブリッジは、一方のバスからのトランザクションのスプリット識別子を所定ビット数のブリッジトランザクション識別子に変換し、このブリッジトランザクション識別子と当該バスブリッジのブリッジ識別子とを組み合わせることにより、他方のバスへ発行するトランザクションのスプリット識別子を生成する。ここで、ブリッジトランザクション識別子のビット数は、スプリット識別子のビット数からブリッジ識別子のビット数を減算した値である。なお、この構成においては、一つのバスブリッジに対して上位バスに対するブリッジ識別子と下位バスに対するブリッジ識別子が与えられ、各バスに対するブリッジ識別子としては、当該バスにおいて当該バスブリッジを一意的に識別できる値が与えられる。このような構成によれば、下位バスにもプロセッサを接続することが可能になる。

【００３０】なお、この構成において、さらにスプリット識別子プロセッサ識別子とブリッジ識別子を同ビット数にすれば、プロセッサとバスブリッジとを識別子に関して等価として扱うことができるので、下位バスにプロセッサを接続したり、上位バスにバスブリッジを接続したりすることができ、システム構成の柔軟性を向上させることができる。また、さらに、バスブリッジが付与可能なブリッジトランザクション識別子が現在使用されているか否かを管理する管理テーブルを用意し、この管理テーブルを用いてブリッジトランザクション識別子を決定することにより、ブリッジトランザクション識別子を無駄なく割り当てることができる。

【００３１】また、本発明は、階層バスシステムにおいてバスの物理的仕様及び論理的仕様を下位バスと上位バスとで共通としたことを特徴とする。

【００３２】この構成によれば、上位バスと下位バスとの仕様を同一にすることにより、プロセッサ及びバスブリッジを上位バス及び下位バスのいずれにも接続することができるようになるので、システム構成のバリエーションを広げることができる。

【００３３】また、本発明は、階層バスシステムにおいて、下位バスを複数設け、これら各下位バスにそれぞれメインメモリを接続し、各上位バスに対して下位バスと同数のバスブリッジを接続し、前記各上位バスと各下位バスとを互いに１つのバスブリッジを介して接続したことを特徴とする。

【００３４】この構成では、１つのプロセッサは、複数

のバスブリッジを介して複数のメインメモリに接続される。各メインメモリはそれぞれ別々の下位バスに接続されているので、複数のメインメモリの総メモリサイズと同じメモリサイズの一つのメインメモリを採用するシステム構成をとった場合に比べて、下位バスの付加を低減することができ、下位バスのスループットを向上させることができる。なお、この構成において、複数の下位バスのそれぞれに対してバスブリッジを設け、これらバスブリッジを介して前記各下位バスを１つのＩ／Ｏバスに接続し、このＩ／ＯバスにＩ／Ｏ装置を接続すれば、必要なＩ／Ｏ装置を各下位バスごとに設ける必要がなくなり、システムの大規模化を防止することができる。

【００３５】

【発明の実施の形態】以下、本発明に係るマルチプロセッサシステムの好適な実施形態を図面に基づいて説明する。

【００３６】実施形態１．図１は、本発明に係るマルチプロセッサシステムの全体的な構成の一例を示す概略構成図である。

【００３７】図１において、ＣＰＵ１０～１５は、２つのグループ（ＣＰＵ１０～１２，ＣＰＵ１３～１５）にグループ分けされ、各グループごとに１本の上位バス２０、２２に接続されている。

【００３８】各上位バス２０、２２は、それぞれバスブリッジ３０、３２に接続されている。各バスブリッジ３０、３２は、それぞれキャッシュ（以下、ブリッジキャッシュと呼ぶ）３１、３３を有している。ブリッジキャッシュ３１は、上位バス２０上のＣＰＵ１０～１２によって共有され、ブリッジキャッシュ３３は、上位バス２２上のＣＰＵ１３～１５によって共有される。このように、複数のＣＰＵをグループ化して１つのバスブリッジ（及びブリッジキャッシュ）に接続したものをクラスタと呼ぶ。各バスブリッジ３０、３２は、下位バス４０によって相互接続され、この下位バス４０にメインメモリ５０及びＩ／Ｏ装置５２が接続される。なお、Ｉ／Ｏ装置５２は１つとは限らず、複数接続することもできる。

【００３９】したがって、例えば、ＣＰＵ１１から上位バス２０に発行されたトランザクションがリード（Ｒｅａｄ）であった場合には、バスブリッジ３０は、ブリッジキャッシュ３１をスヌープ（検索）し、この結果ブリッジキャッシュ３１にヒットした場合は検索されたデータを上位バス２０に返し、ヒットしなかった場合には下位バス４０にリード・トランザクションを発行する。

【００４０】図１の構成において、上位バス２０及び２２はスプリットバスであり、ＣＰＵから発せられるメモリトランザクション（メインメモリのデータ内容についてのトランザクション）及びＩ／Ｏトランザクション（Ｉ／Ｏ装置に対するトランザクション）の両方についてスプリット可能となっている。また、ＣＰＵ１０～１５は、スプリット対応可能なＣＰＵである。トランザクションをスプリットをするかしないかの判断はバスブリッジ３０及び３２で行われる。

【００４１】ここで、トランザクションをスプリットした場合におけるシステム各部の動作について説明する。上位バスに発せられたトランザクションをスプリットすると決定した場合、バスブリッジは、トランザクション発行元のＣＰＵに対して、当該トランザクションをスプリットする旨の応答（「スプリット終了」と呼ぶ）を返すとともに、下位バスに対して必要なトランザクションを発行する。一方、スプリット終了信号を受け取った要求元ＣＰＵは、いったん上位バスを解放し、下位バスからの応答を待つ。バスブリッジは、下位バスに発行したトランザクションに対する他のバスブリッジやメインメモリなどからの応答が揃ったところで、トランザクション発行元ＣＰＵに対するスプリット応答トランザクションを生成し、上位バス上に発行する。そして、トランザクション発行元ＣＰＵがこのスプリット応答トランザクションを受け取ったところで、当該ＣＰＵが最初に発行したトランザクションに対する一連の処理が終了する。このように、本構成によれば、上位バスをスプリットバスとしたことにより、トランザクションをスプリットしてからスプリット応答トランザクションが発行されるまでの間は上位バスが解放されているので、その間上位バスに他のトランザクションを発行することができる。

【００４２】このように、図１の構成によれば、まずＣＰＵ１０～１５とメインメモリ５０、Ｉ／Ｏ装置５２との間が階層的なバス構成となっているため、個々の上位バス２０、２２に接続できるＣＰＵの数が少なくても、システム全体として多数のＣＰＵを接続することができる。

【００４３】また、この構成では、ブリッジキャッシュ３１、３３を設けたことにより、上位バス２０、２２のトランザクションの要求データがブリッジキャッシュに存在する場合には、メインメモリ５０を読みに行く必要がなくなるため、下位バス４０のバス負荷が低減され、この結果下位バス４０のスループットが向上する。また、ＣＰＵが自らの発したトランザクションに対するレスポンスを得るまでのレイテンシも短くなる。

【００４４】さらに、この構成では、上位バス２０、２２をスプリットバスとしたことにより、上位バスのスループットを向上させることができる。すなわち、この構成では、上位バス２０、２２のトランザクションを下位バス４０に転送する必要がある場合には、そのトランザクションをスプリットすることができるので、その間上位バス２０、２２に他のトランザクションを発行することができ、全体として上位バスのスループットを向上させることができる。

【００４５】なお、図１の構成はあくまで一例であり、上位バス及び下位バスには、物理的制約を満たす範囲内であれば、ＣＰＵ及びバスブリッジをいくつでも接続す

ることができる。

【００４６】さて、図１の構成において、ＣＰＵ１０～１５にそれぞれ専用キャッシュ（以下、ＣＰＵキャッシュと呼ぶ）を設けた場合を考える。このように各ＣＰＵに専用のＣＰＵキャッシュを設ければ、上位バス２０、２２に発行されるトランザクションが少なくなるので、上位バス２０、２２のスループットを向上させることができる。ただし、この場合には、ＣＰＵキャッシュ、ブリッジキャッシュ、及びメインメモリの間のデータの一貫性（キャッシュ・コンシステンシ）を保つ必要がある。

【００４７】キャッシュ・コンシステンシを維持するためのプロトコルとしては、従来より様々な方式が提案されているが、その代表的なものにＭＥＳＩプロトコルがある。ＭＥＳＩプロトコルは、ライトバック・キャッシュのためのキャッシュ・コンシステンシ・プロトコルの一つであり、各キャッシュブロックに対してＭ，Ｅ，Ｓ，Ｉの４種類のいずれかのステート（状態）を与え、キャッシュブロックの読み込み、書き換えの際にそのステートを一定の規則にしたがって制御することによりキャッシュ・コンシステンシを維持する。

【００４８】ここで、Ｍ，Ｅ，Ｓ，Ｉの各ステートの示す意味は以下の通りである。

【００４９】Ｍ(Modify)：そのブロックの更新されたデータ（すなわち、そのブロックの最新のデータ）をそのキャッシュだけが持つ。

【００５０】Ｅ(Exclusive)：そのブロックに関しメインメモリと同一内容のデータをそのキャッシュだけが持つ。

【００５１】Ｓ(Shared)：そのブロックに関しメインメモリと同一内容のデータを持つが、他のキャッシュも当該データを持っている可能性がある。

【００５２】Ｉ(Invalid)：そのブロックは無効である。

【００５３】キャッシュブロックの読み込み、書き換えなどの操作を行った場合のステートの制御については、ＭＥＳＩプロトコルに規則が定められている。例えば、あるＣＰＵのキャッシュのステートＳのキャッシュブロックにライト（書込み）が行われる場合について説明する。この場合、他のＣＰＵのキャッシュが当該キャッシュブロックをステートＳで持っている可能性がある。したがって、ライトを行うＣＰＵは、他のキャッシュの当該キャッシュブロックを無効化するため、インバリデート（Invalidate: 無効化）・トランザクションを発行する。これを受けた他のＣＰＵは、自らのキャッシュの当該キャッシュブロックのステートをＩとし、ライトを行うＣＰＵ自体は所望のライト処理を行ったのち当該キャッシュブロックのステートをＭに変更する。このような操作により、ＣＰＵがキャッシュ内の古いデータを使用するおそれが無くなり、キャッシュ・コンシステンシが保たれる。すなわち、あるキャッシュがステートＳのブロックを持っている場合において当該ブロックについてライト（データの書換え）を行う場合には、キャッシュ・コンシステンシ維持のためには、他のキャッシュの当該ブロックを無効化する。

【００５４】また、別の例としては、あるキャッシュがステートＥのブロックを持っている場合において当該ブロックに対してライトを行う場合には、他のキャッシュはすべて当該ブロックを持っていない（ステートＩ）ので、他のキャッシュに対して何ら手当てをすることなくライト処理を行うことができる。ＭＥＳＩプロトコルには、その他詳細な規則が定められているが、ここでは省略する。

【００５５】本実施形態では、基本的にこのＭＥＳＩプロトコルを用いてキャッシュ・コンシステンシの管理を行うが、ＭＥＳＩプロトコル自体は１階層のキャッシュシステムについて規定したものであり、本実施形態のようなキャッシュが複数階層構成となっているものに対しては、そのままで利用することはできない。そこで本実施形態では、上記ＭＥＳＩプロトコルを次のように拡張する。

【００５６】すなわち、この拡張プロトコルでは、ＭＥＳＩの各ステートは、同レベルにあるキャッシュ同士の関係を示すものと規定する。ＣＰＵキャッシュのステートは他のＣＰＵキャッシュとの関係を示すものとし、ブリッジキャッシュのステートは他のブリッジキャッシュとの関係を示すものとする。この場合、ＣＰＵキャッシュのステートは前述したものと同じ意味となるが、ブリッジキャッシュのステートの意味は以下に示す通りとなる。

【００５７】Ｍ：そのブロックの更新されたデータ（すなわち、そのブロックの最新のデータ）を、そのブリッジキャッシュ自体又はそのブリッジキャッシュの上位のＣＰＵキャッシュが持つ。

【００５８】Ｅ：そのブリッジキャッシュは、そのブロックに関しメインメモリと同一内容のデータを有しており、他のブリッジキャッシュは当該ブロックのデータを有しない。

【００５９】Ｓ：そのブロックに関しメインメモリと同一内容のデータを持つが、他のブリッジキャッシュも当該データを持っている可能性がある。

【００６０】Ｉ：そのブロックは無効である。

【００６１】なお、ここで、「ブリッジキャッシュの上位のＣＰＵキャッシュ」とは、あるブリッジキャッシュに対し、バスブリッジ及び上位バス経由で接続されたＣＰＵのキャッシュのことを示すものとする。逆に「ＣＰＵキャッシュの下位のブリッジキャッシュ」とは、あるＣＰＵキャッシュに対し、上位バス及びバスブリッジを経由して接続されたブリッジキャッシュのことを示すものとする。また、「キャッシュのステート」と言った場

合、そのキャッシュにおける、現在注目しているキャッシュブロック（すなわち、現在のトランザクションの要求アドレスに対応するキャッシュブロック）のステートのことを示すものとする。

【００６２】そして、本実施形態では、ブリッジキャッシュのステートとＣＰＵキャッシュのステートとの関係に一定の制約を与えることにより、ブリッジキャッシュとＣＰＵキャッシュとのマルチレベル包含性（ＭＬＩ）を保証し、多階層のキャッシュシステムにおけるキャッシュ・コンシステンシを維持する。ここで、ＭＬＩとは、多階層のキャッシュシステムにおけるキャッシュ・コンシステンシの制御を容易ならしめるために採用される制約の一種であり、ＣＰＵキャッシュに存在するデータは必ずその下位のブリッジキャッシュに存在するという性質のことをいう。なお、本実施形態においては、ＭＬＩを採用するため、ブリッジキャッシュの容量は、少なくともその上位のＣＰＵキャッシュの容量の総和以上となっている。ＭＬＩを維持するようにキャッシュを制御するようにすれば、システムを簡単な回路で構成することができ、余計なバス・トランザクションの発生を抑制することができる。

【００６３】本実施形態では、ＭＬＩを維持するために、ブリッジキャッシュのステートに基づきＣＰＵキャッシュの取り得るステートに制約を与える。本実施形態では、この制約として、以下に示す２種類の制約のいずれかを採用する。

【００６４】まず、第１の制約（以下、「制約１」と呼ぶ）を表１に示す。

【００６５】

【表１】

ブリッジキャッシュ	ＣＰＵキャッシュ
M	M，E，S，I
E	S，I
S	S，I
I	I

次に、第２の制約（以下、「制約２」と呼ぶ）を表２に示す。

【００６６】

【表２】

ブリッジキャッシュ	ＣＰＵキャッシュ
M	M，E，S，I
E	M，E，S，I
S	S，I
I	I

表１及び表２は、ブリッジキャッシュのステートがM，Ｅ，Ｓ，Ｉのそれぞれの場合におけるそのブリッジキャッシュの上位のＣＰＵキャッシュが取り得るステートの種類を示している。

【００６７】制約１及び制約２は、ブリッジキャッシュのステートがM，Ｓ，Ｉの時については共通である。

【００６８】制約１及び制約２では、ブリッジキャッシュのステートがＩである場合にＣＰＵキャッシュのステートはＩしか認めないことにより、ブリッジキャッシュに無いデータ（キャッシュブロック）はその上位のリーダーＣＰＵキャッシュにも無いということを保証する。

【００６９】また、制約１及び制約２では、ブリッジキャッシュのステートがＳである場合には、ＣＰＵキャッシュのステートにＳ又はＩを認める。前述したように、ブリッジキャッシュのステートがＳとなるのは、ブリッジキャッシュが下位バスから新たなデータを読み込んだときに他のブリッジキャッシュが同一データをもっている場合である。ここで、ＣＰＵキャッシュのステートを制約１又は２のごとく制限しておけば、ＣＰＵキャッシュがそのデータをブリッジキャッシュから読み込んだときにＣＰＵキャッシュのステートがＳとなる。これにより、次にそのＣＰＵキャッシュにおいてそのデータの書換え（ライト）が行われる場合には、必ず他のキャッシュの無効化のためのトランザクション（インバリデート：Invalidate）が発行される。そして、そのインバリデートをそのＣＰＵの下位のバスブリッジが受け取ってそのバスブリッジから下位バスにもインバリデートを発行することにより、他のブリッジキャッシュ（及びその上位のＣＰＵキャッシュ）にある当該データを無効化することができる。

【００７０】また、制約１及び制約２では、ブリッジキャッシュのステートがMである場合にＣＰＵキャッシュのステートとしてM，Ｅ，Ｓ，Ｉのすべてのステートを認める。本実施形態において、ブリッジキャッシュのステートがMになるのは、上位のＣＰＵにおいてライト動作が行われた場合である。したがって、この場合は、ＣＰＵキャッシュにはすべてのステートを認めておく必要がある。

【００７１】制約１と制約２の相違点は、ブリッジキャッシュがＥのときに制約２ではＣＰＵキャッシュのステートとしてM，Ｅを認めるが、制約１ではM，Ｅを認めないという点である。

【００７２】ブリッジキャッシュがＥとなるのは、ブリッジキャッシュが下位バスから新たなデータを読み込んだときに他のブリッジキャッシュが同一データをもっていない場合である。この場合において、あるＣＰＵキャッシュがブリッジキャッシュからそのデータを読み込んだ場合、制約２では同一上位バス上の他のＣＰＵキャッ

シュがそのデータを持っていない場合には、そのＣＰＵキャッシュのステートをＥとすることを認める。これに対して制約１では、同じ状況でそのＣＰＵキャッシュのステートをＳとする。

【００７３】制約２では、ＣＰＵキャッシュがあるデータをステートＥで持っている場合においてそのデータを書き替えるときには、そのＣＰＵは外部に何ら信号を発せずに書き替えを実行し、その結果ＣＰＵキャッシュのステートがＭに変わる。この制約２では、ブリッジキャッシュのステートがＥである場合に、その上位のＣＰＵが最新のデータを持っているかどうかを知るには、上位バスをスヌープするなどの処理が必要がある。

【００７４】一方、制約１では、ブリッジキャッシュがあるデータをステートＥで持っている場合に、その上位のＣＰＵキャッシュが当該データを持っている場合にはそのステートはＳになる。したがって、ＣＰＵキャッシュが当該データを書き替える場合には上位バス上に必ずインバリデートが発行されるため、ブリッジキャッシュはそのインバリデートを受け取ることにより、ＣＰＵキャッシュのデータが書き替えられたことを知ることができる。このときブリッジキャッシュのステートはＭに変わる。したがって、制約２によれば、ブリッジキャッシュのステートがＥのときには、その上位のＣＰＵキャッシュには最新のデータが存在しないことが保証される。

【００７５】本実施形態では、このように、ブリッジキャッシュとその上位のＣＰＵキャッシュとのステートの関係に制約を与えることにより、無用なバス・トランザクションを増やさずにキャッシュ・コンシステンシを維持することができる。

【００７６】［バスブリッジの動作］本実施形態では、バスブリッジの働きにより、以上説明した制約を満たし、キャッシュ・コンシステンシを維持する。以下、本実施形態のバスブリッジ及びブリッジキャッシュの動作について説明する。

【００７７】［１］上位バスに発行されたトランザクションに対する基本動作

まず、ＣＰＵから上位バスに発行されたトランザクションを受け取った場合のバスブリッジの一連の動作について説明する。

【００７８】本実施形態においては、上位バスには以下のトランザクションが定義されている。

【００７９】（１）Ｒｅａｄ（リード）：アドレスで示されたキャッシュブロックを読み込む。ＣＰＵからリードのリクエストが出され、ＣＰＵキャッシュでミスした場合（すなわち、リード・リクエストの要求データがＣＰＵキャッシュ内になかった場合）に発行される。

【００８０】（２）Ｉｎｖａｌｉｄａｔｅ（インバリデート）：他のＣＰＵキャッシュに存在するアドレスで示されたキャッシュブロックを無効化する。ＣＰＵからライトのリクエストが出され、ＣＰＵキャッシュでヒットしたもののそのＣＰＵキャッシュのステートがＳであった場合（ライト・リクエストの要求データがＣＰＵキャッシュ内にステートＳで存在した場合）に発行される。

【００８１】（３）Ｒｅａｄ＆Ｉｎｖａｌｉｄａｔｅ（リード・アンド・インバリデート）：他のＣＰＵキャッシュに存在するアドレスで示されたキャッシュブロックを無効化し、当該アドレスで示されたキャッシュブロックを読み込む。ＣＰＵからライト・リクエストが出され、ＣＰＵキャッシュでミスした場合に発行される。

【００８２】（４）ＷｒｉｔｅＢａｃｋ（ライトバック）：アドレスで示されたキャッシュブロックを下位のブリッジキャッシュに書き戻す。ＣＰＵのリクエストによりＣＰＵキャッシュに空きブロックを作る必要があるときに発行される。

【００８３】なお、以上は、メインメモリのデータに対するトランザクション（メモリ・トランザクション）のみである。このほかにもＩ／Ｏ装置に対するトランザクション（Ｉ／Ｏトランザクション）があるが、これについての説明は省略する。

【００８４】［１．１］下位バス出力判定

上位バスにトランザクションが発行された場合、バスブリッジはまずそのトランザクションをスプリットする必要がないか判定し、スプリットする場合には下位バスに発行するトランザクションの種類を決定する。

【００８５】ここで、下位バスには、次に示すトランザクションが定義されている。

【００８６】（１）Ｒｅａｄ（リード）：指定されたアドレスのキャッシュブロックを読み込む。

【００８７】（２）Ｉｎｖａｌｉｄａｔｅ（インバリデート）：他のブリッジキャッシュに存在する指定されたアドレスのキャッシュブロックを無効化する。

【００８８】（３）Ｒｅａｄ＆Ｉｎｖａｌｉｄａｔｅ（リード・アンド・インバリデート）：他のブリッジキャッシュに存在する指定されたアドレスのキャッシュブロックを無効化し、当該アドレスで示されたキャッシュブロックを読み込む。

【００８９】（４）ＷｒｉｔｅＢａｃｋ（ライトバック）：指定されたアドレスのキャッシュブロックを下位のメインメモリに書き戻す。

【００９０】なお、下位バスにも上位バス同様Ｉ／Ｏトランザクションが定義されているが、ここでは省略する。

【００９１】［１．１．１］制約１を採用する場合

制約１を採用する場合には、バスブリッジは表３に示す規則にしたがってスプリットの要否の判定及び下位バスに発行するトランザクションを決定する。

【００９２】

【表３】

	上位バス・トランザクション			
ブリッジキャッシュステート	Read	Read &Invalidate	Invalidate	WriteBack
M	○	○	○	○
E	○	○	○	×
S	○	Invalidate	Invalidate	×
I	Read	Read&Invalidate	×	×

○：下位バスにトランザクションを発行しない
×：あり得ない

表３に示すように、バスブリッジは、自らの上位バスに発行されたトランザクションの種類とそのトランザクションを受け取ったときのブリッジキャッシュのステートとに基づき、スプリットするか否か及びスプリットする場合における下位バスに発行するトランザクションの種類を決定する。なお、以下においては、「ブリッジキャッシュがＸ（ＸはＭ、Ｅ、Ｓ、Ｉのうちのどれかのステート）である」とは、バスブリッジがトランザクションを受け取った場合において、そのトランザクションのアドレスに対応する当該ブリッジキャッシュのブロックのステートがＸであることを意味する。

【００９３】表３に示すように、上位バスにＲｅａｄが発行された場合、バスブリッジは、ブリッジキャッシュがＩの場合にのみ、上位バスのＲｅａｄをスプリットして下位バスにＲｅａｄを発行する。なお、ブリッジキャッシュがＭ，Ｅ，Ｓのいずれかである場合は、Ｒｅａｄトランザクションの要求アドレスはブリッジキャッシュ又は同一上位バス上のＣＰＵキャッシュに存在するので、要求元ＣＰＵはそれらから要求データを得ることができる。したがって、この場合は下位バスにはトランザクションが発行されない。なお、バスブリッジが上位バスのＲｅａｄをスプリットしなかった場合には、上位バスだけでトランザクションが完了し、ブリッジキャッシュのステートはトランザクションの前と変化しない。

【００９４】上位バスにＲｅａｄ＆Ｉｎｖａｌｉｄａｔｅが発行された場合は、バスブリッジは、ブリッジキャッシュがＳ又はＩのときにトランザクションをスプリットする。まず、ブリッジキャッシュがＳの場合には、バスブリッジは上位バスのＲｅａｄ＆Ｉｎｖａｌｉｄａｔｅをスプリットし、下位バスに対してＩｎｖａｌｉｄａｔｅを発行する。この場合は、Ｒｅａｄ＆Ｉｎｖａｌｉｄａｔｅで要求されるデータはブリッジキャッシュに存在するので、バスブリッジは、当該データをスプリット終了信号と共に要求元ＣＰＵに返し、他のバスブリッジのブリッジキャッシュを無効化するために下位バスにＩｎｖａｌｉｄａｔｅを発行する。一方、ブリッジキャッシュのがＩの場合には、バスブリッジはトランザクションをスプリットし、下位バスに対してＲｅａｄ＆Ｉｎｖａｌｉｄａｔｅを発行する。この場合は、Ｒｅａｄ＆Ｉｎｖａｌｉｄａｔｅで要求されるデータはブリッジキャッシュに存在しないので、バスブリッジは、当該データを入手しかつ他のバスブリッジのブリッジキャッシュを無効化するため、下位バスにＲｅａｄ＆Ｉｎｖａｌｉｄａｔｅを発行する。なお、ブリッジキャッシュがＭ又はＥの場合は、バスブリッジはトランザクションをスプリットしない。ステートがＭ又はＥということは、当該ブリッジキャッシュ又はその上位のＣＰＵキャッシュに要求データが存在し、他のブリッジキャッシュ及びその上位のＣＰＵキャッシュにはその要求データが存在しないからである。なお、ブリッジキャッシュがＭ又はＥの場合は、トランザクション終了後のブリッジキャッシュのステートは共にＭとなる。

【００９５】上位バスにＩｎｖａｌｉｄａｔｅが発行された場合は、バスブリッジは、ブリッジキャッシュがＳの場合に、上位バスのＩｎｖａｌｉｄａｔｅをスプリットし、下位バスに対してＩｎｖａｌｉｄａｔｅを発行する。ブリッジキャッシュがＳであるため、他のブリッジキャッシュが同じデータを持っている可能性があるからである。そして、ブリッジキャッシュのステートがＭ又はＥの場合は、バスブリッジはトランザクションをスプリットしない。この場合、他のブリッジキャッシュは無効化対象のデータ（ブロック）を有していないからである。なお、ブリッジキャッシュがＩである場合は、ＣＰＵから上位バス上にＩｎｖａｌｉｄａｔｅが発行されることはない。ブリッジキャッシュがＩの場合その上位のＣＰＵキャッシュのステートは必ずＩであるが、Ｉｎｖ

ａｌｉｄａｔｅはＣＰＵキャッシュがステートＳのときにライト動作を行う場合にしか発行されないからである。なお、上位バスに発行されたＩｎｖａｌｉｄａｔｅが終了した後は、ブリッジキャッシュのステートはＭとなる。

【００９６】また、上位バスに発行されたトランザクションがＷｒｉｔｅＢａｃｋであった場合は、トランザクションのスプリットは行われない。ＷｒｉｔｅＢａｃｋトランザクションでは、ＣＰＵキャッシュにある最新データをブリッジキャッシュに書き戻すだけなので、他のバスブリッジには何ら影響がないからである。なお、ＷｒｉｔｅＢａｃｋが発行されるのはＣＰＵキャッシュがＭのブロックを持っている場合だけであり、制約１ではＣＰＵキャッシュがＭとなり得るのは下位のブリッジキャッシュがＭの場合だけである。したがって、ブリッジキャッシュがＥ，Ｓ，Ｉのときは上位バス上にＷｒｉｔｅＢａｃｋが発行されることはない。

【００９７】以上説明した規則にしたがってバスブリッジが下位バスに発行するトランザクションを決定することにより、表１に示したブリッジキャッシュとＣＰＵキャッシュとの制約１を満足させ、キャッシュ・コンシステンシを維持することができる。

【００９８】そして、バスブリッジは、このようにして決定されたトランザクションを下位バスに発行する。この下位バスに発行されたトランザクションを受け取った他のバスブリッジの動作については後の［２］節において説明する。

【００９９】次に、以上説明した規則の変形例について説明する。変形例では、上位バスのトランザクションの要求アドレスが、命令コードを指すものか、それともいわゆる「データ」（非命令コード）を指すものかを判定し、この判定を下位バスに発行するトランザクションの種類の決定に反映させる。ここで、Ｉｎｖａｌｉｄａｔｅ及びＲｅａｄ＆Ｉｎｖａｌｉｄａｔｅ（以下、Ｉｎｖａｌｉｄａｔｅ系と呼ぶ）は、ＣＰＵがライトを行う場合に発行されるものなので、命令コードを要求することはほとんどない。したがって、この変形例では、Ｒｅａｄについてのみ、命令コードを要求する場合とデータを要求する場合とでスプリット時に下位バスに発行するトランザクションの種類を変える。この変形例における規則を表４に示す。

【０１００】

【表４】

ブリッジキャッシュステート	上位バス・トランジクション				
	Read		Read &Invalidate	Invalidate	WriteBack
	命令コード	データ			
M	○	○	○	○	○
E	○	○	○	○	×
S	○	○	Invalidate	Invalidate	×
I	Read	Read & Invalidate	Read&Invalidate	×	×

○：下位バスにトランザクションを発行しない

×：あり得ない

表４は、Ｒｅａｄの欄以外は表３と全く同じである。表４から分かるように、この変形例では、上位バスに発行されたＲｅａｄをスプリットするときに、そのＲｅａｄトランザクションが命令コードを要求するものである場合には下位バスにＲｅａｄを発行し、データを要求するものである場合には下位バスにＲｅａｄ＆Ｉｎｖａｌｉｄａｔｅを発行する。

【０１０１】すなわち、この変形例では、上位バスのＲｅａｄがデータ（非命令コード）を要求する場合は、スプリットの際にバスブリッジは、下位バスにＲｅａｄ＆Ｉｎｖａｌｉｄａｔｅを発行することにより、当該データを得ると共に他のブリッジキャッシュの当該データのブロックを無効化する。これにより、バスブリッジは、下位バスから得られた当該データをブリッジキャッシュにステートＭで登録することが可能となる。なお、当該データを読み込んだときにブリッジキャッシュのステートをＭにするための機構については後のブリッジキャッシュの次ステートの決定方法の説明において説明する。

【０１０２】命令コードは、読み込まれた後に書き替えられることはないが、データ（非命令コード）は、読み込まれた後に書き替えられる可能性が高い。キャッシュ内のデータを書き替える場合、そのデータが含まれるキ

ャッシュブロックのステートがＳであれば、Ｉｎｖａｌｉｄａｔｅを発行して他のキャッシュの当該ブロックを無効化する必要がある。したがって、データをブリッジキャッシュに読み込んだときにステートＳで登録したとすると、そのデータが書き替えられるときにバスブリッジから下位バスにＩｎｖａｌｉｄａｔｅを発行する必要が出てくる。この場合、データの読み込みのときに１回（Ｒｅａｄ）、データの書き替えのときに１回（Ｉｎｖａｌｉｄａｔｅ）の合計２回、下位バスにトランザクションが発行されることになる。これに対して、本変形例では、書き替えられる可能性が高いデータをブリッジキャッシュに読み込む必要があるときには、下位バスにＲｅａｄではなくＲｅａｄ＆Ｉｎｖａｌｉｄａｔｅを発行することにより予め他のブリッジキャッシュを無効化し、そのデータをステートＭでブリッジキャッシュに登録することにより、データの書き替えのときには下位バスにトランザクションを発行する必要がなくなる。したがって、本変形例では、データを読み込んだあとそのデータに書き替えを行う場合、読み込みの時点で下位バスにＲｅａｄ＆Ｉｎｖａｌｉｄａｔｅを１回発行するのみでよく、上記表３の場合に比べて下位バスに発行するトランザクションの回数を減らすことができる。データ（非命令コード）は書き替えられる可能性は高いので、本変形例の手法による下位バス負荷低減の効果は高い。

【０１０３】なお、本変形例では、上位バスのＲｅａｄが命令コードを要求するものであった場合は、表３の場合と同様、スプリット時には下位バスにＲｅａｄを発行する。これにより、下位バスから得られた命令コードはバスブリッジにステートＥ又はＳで登録される。

【０１０４】［１．１．２］制約２を採用する場合

制約２を採用する場合、バスブリッジは、上位バスに発行されたトランザクションを受け取ると、そのトランザクションの種類とそのトランザクションを受け取ったときのブリッジキャッシュのステートとに基づき、表５に示す規則にしたがってスプリットの要否の判定及び下位バスに発行するトランザクションを決定する。

【０１０５】

【表５】

	上位バス・トランザクション			
ブリッジキャッシュステート	Read	Read &Invalidate	Invalidate	WriteBack
M	○	○	○	○
E	○	○	○	○
S	○	Invalidate	Invalidate	×
I	Read	Read&Invalidate	×	×

○：下位バスにトランザクションを発行しない

×：あり得ない

表５の規則と前記制約１に対する表３の規則との違いは、ブリッジキャッシュのステートがＥのときに上位バスにＷｒｉｔｅＢａｃｋが発行される可能性があるか否かの違いである。制約１では、ブリッジキャッシュのステートがＥの場合、ＣＰＵキャッシュにステートＥを認めず、したがってＣＰＵキャッシュがステートＭとなることはない（表１参照）ので、ＣＰＵからＷｒｉｔｅＢａｃｋが行われることはない。これに対して、制約２では、ＣＰＵキャッシュにステートＥを認めており（表２参照）、ＣＰＵキャッシュのステートＥのブロックがＣＰＵからの書き換えにより、バスブリッジが関知しないままステートＭとなっている可能性がある。したがって、制約２では、ブリッジキャッシュがステートＥのときに、上位のＣＰＵキャッシュからＷｒｉｔｅＢａｃｋが行われる可能性がある。

【０１０６】表５の規則と前記表３の規則との相違はこの点のみであり、上位バス・トランザクションのスプリットの決定及び下位バスに発行するトランザクションの決め方については、表５の規則は表３の規則と同一である。

【０１０７】以上説明した規則にしたがってバスブリッジが下位バスに発行するトランザクションを決定することにより、表２に示したブリッジキャッシュとＣＰＵキャッシュとの制約２を満足させ、キャッシュ・コンシステンシを維持することができる。

【０１０８】［１．２］ブリッジキャッシュの次ステート

次に、上位バスからのトランザクションによるブリッジキャッシュのステート遷移について、その上位バス・トランザクションをスプリットしない場合とスプリットする場合に分けて説明する。なお、ブリッジキャッシュの次ステートの決定の仕方は、制約１及び制約２のいずれを採用した場合も同じである。

【０１０９】まず、スプリットしない場合について説明

する。ここで、上位バス・トランザクションがＲｅａｄ又はＷｒｉｔｅＢａｃｋであった場合には、ブリッジキャッシュのステートは当該トランザクションの前後で変化しない。一方、上位バス・トランザクションがＩｎｖａｌｉｄａｔｅ系であった場合には、当該トランザクション終了後のブリッジキャッシュのステートはＭとする。Ｉｎｖａｌｉｄａｔｅ系トランザクションはデータの書き替えのために発行されるからである。

【０１１０】次に、スプリットする場合について説明する。この場合、トランザクション終了後のブリッジキャッシュのステート（以下、次ステートと呼ぶ）は、バスブリッジから下位バスに発行したトランザクションとこのトランザクションに対する下位バスのスヌープ結果に基づいて決定する。

【０１１１】表６は、ブリッジキャッシュの次ステートの決定規則を示す表である。

【０１１２】

【表６】

	下位バスのスヌープ結果		
下位バスに発行したトランザクション	ミス	ＨＩＴ	ＨＩＴＭ
Read Invalidate Read&Invalidate	E M M	S M M	S M M

ここで、スヌープ結果について説明する。バス上にトランザクションが発行された場合、そのバスに接続されたエージェント（ＣＰＵやバスブリッジ等）は、そのトランザクションの要求アドレスについて、最新データ（すなわち、ステートＭ）を持っているか、メモリと同内容のデータ（ステートＥ又はＳ）を持っているか、を示す信号をバス上に出力する。この信号をスヌープ出力という。以下、簡単のため、最新データを持っていることを示す信号をＨＩＴＭ、メモリと同内容のデータを持っていることを示す信号をＨＩＴという。より詳しく説明すれば、本実施形態では、バスにはＨＩＴＭ用の信号線とＨＩＴ用の信号線が設けられており、トランザクションを受け取ったエージェントは、要求されたブロックをステートＭで持っている場合にはＨＩＴＭをアサートし、要求されたブロックをステートＥ又はＳで持っている場合にはＨＩＴをアサートする。そして、エージェントがトランザクションの要求アドレスに対応するブロックを持っていない場合（ステートＩのとき）には、ＨＩＴＭもＨＩＴもアサートされず、この状態を「ミス」と呼ぶ。

【０１１３】トランザクション発行元のエージェントは、他のエージェントからのスヌープ出力のワイヤードＯＲをスヌープ結果として受け取る。したがって、スヌープ結果がＨＩＴＭの場合は、バス上に当該トランザクションの要求アドレスに対応するブロックをステートＭで持っているエージェントが存在することを意味し、スヌープ結果がＨＩＴの場合は、バス上に当該トランザクションの要求アドレスに対応するブロックをステートＥ又はＳで持っているエージェントが存在することを意味する。そして、スヌープ結果がミスの場合、すなわちＨＩＴＭでもＨＩＴでもない場合は、バス上に当該トランザクションの要求アドレスに対応するブロックを持っているエージェントがないこと意味する。

【０１１４】表６に示すように、バスブリッジが下位バスにＲｅａｄを発行した場合には、下位バスのスヌープ結果によってブリッジキャッシュの次ステートを変える。すなわち、下位バススヌープ結果がＨＩＴ又はＨＩＴＭであったときは次ステートをＳとし、下位バススヌープ結果がミスであったときは次ステートをＥとする。一方、バスブリッジが下位バスにＩｎｖａｌｉｄａｔｅ系トランザクションを発行した場合には、下位バスのスヌープ結果によらず次ステートをＭとする。

【０１１５】以上説明したようにブリッジキャッシュの次ステートを決めることにより、ブリッジキャッシュ同士の間のキャッシュ・コンシステンシを維持することができる。

【０１１６】［１．３］スプリット応答時における上位バスへのスヌープ出力

バスブリッジが上位バスに発行されたトランザクションをスプリットした場合、そのバスブリッジは、スプリット時に下位バスに発行したトランザクションに対する他の下位バスエージェントからの応答を受け取り、その結果に基づきトランザクション発行元のＣＰＵに対して応答を行う必要がある。バスブリッジからトランザクション発行元ＣＰＵへのこのような応答のことをスプリット応答トランザクションという。バスブリッジは、下位バスエージェントからの応答が揃ったところで上位バスにアービトレーションを行って上位バスの使用権を獲得し、上位バスに対し必要な情報を含んだスプリット応答トランザクションを発行する。本実施形態では、このスプリット応答トランザクションのときにバスブリッジから上位バスに対して所定の規則にしたがったスヌープ出力を発し、このスヌープ出力により発行元ＣＰＵのキャッシュのステートを制御する。

【０１１７】従来の一階層のキャッシュシステムについてのＭＥＳＩプロトコルでは、ＣＰＵからバス上にトランザクションが発行されたときに、そのバスのスヌープ結果から当該ＣＰＵのキャッシュの次ステートを決定する規則が定められている。その規則によれば、ＣＰＵからバスにＲｅａｄが発行された場合、スヌープ結果がミ

スの場合は当該ＣＰＵキャッシュのステートをＥに変え、スヌープ結果がＨＩＴ又はＨＩＴＭの場合は当該ＣＰＵキャッシュのステートをＳに変える。また、その規則では、ＣＰＵからバスにＩｎｖａｌｉｄａｔｅ系トランザクションが発行された場合は、スヌープ結果にかかわらず当該ＣＰＵキャッシュのステートをＭに変える。

【０１１８】本実施形態では、ＣＰＵキャッシュは従来の一階層キャッシュ用のＭＥＳＩプロトコルにしたがっており、したがってＣＰＵキャッシュのステートはその規則にしたがって遷移する。そして、本実施形態では、その規則を利用してバスブリッジからのスヌープ出力によりその上位のＣＰＵキャッシュのステートを制御することにより、制約１又は制約２の関係を維持する。

【０１１９】［１．３．１］制約１を採用した場合

本実施形態において制約１を採用した場合には、バスブリッジは、スプリットを行った場合に、下位バスに発行したトランザクションの種類に基づき、表７の規則に従って上位バスに発行するスヌープ出力を決定する。

【０１２０】

【表７】

	下位バスのスヌープ結果		
下位バスに発行したトランザクション	ミス	ＨＩＴ	ＨＩＴＭ
Read Invalidate Read&Invalidate	ＨＩＴ（Ｓ） ミス（Ｍ） ミス（Ｍ）	ＨＩＴ（Ｓ） ミス（Ｍ） ミス（Ｍ）	ＨＩＴ（Ｓ） ミス（Ｍ） ミス（Ｍ）

（注）括弧（）はトランザクション発行元のＣＰＵキャッシュの最終ステート

表７において、下位バスに発行したトランザクションがＲｅａｄの場合、バスブリッジは下位バスのスヌープ結果にかかわらず上位バスにスヌープ出力としてＨＩＴをアサートする。下位バス・スヌープ結果がＨＩＴ又はＨＩＴＭの場合は、他のブリッジキャッシュに当該ブロックがあると言うことなので、トランザクション発行元のＣＰＵの次ステートをＳにする必要がある。ここでＣＰＵキャッシュは前述した１階層キャッシュに対するＭＥＳＩプロトコルにしたがってステート遷移するので、次ステートをＳにするにはスヌープ結果としてＨＩＴを受け取る必要がある。そこで、バスブリッジはスプリット応答トランザクション時に上位バスに対してＨＩＴをアサートし、上位バス上のトランザクション発行元ＣＰＵは、そのＨＩＴを受けてＣＰＵキャッシュの対応キャッシュブロックのステートをＳとする。また、表７の規則では、下位バス・スヌープ結果がミスのときにも上位バスにＨＩＴをアサートすることにより、表１に示した制約１を満足するようにしている。すなわち、下位バス・スヌープ結果がミスのときには、他のブリッジキャッシュに当該ブロックを持っているものはないので、ブリッジキャッシュの当該ブロックの次ステートはＥとなる（表６参照）。したがって、この場合に制約１を満たすためには、トランザクション発行元ＣＰＵのキャッシュの次ステートをＳにする必要がある。このため、表７の規則では、下位バス・スヌープ結果がミスの場合にも上位バスに対してＨＩＴをアサートする。

【０１２１】また、下位バスに発行したトランザクションがＩｎｖａｌｉｄａｔｅ系の場合には、バスブリッジは上位バスに対してＨＩＴもＨＩＴＭもアサートしない（すなわち、ミス）。この場合、スプリット前の上位バスのトランザクションもＩｎｖａｌｉｄａｔｅ系であるので、トランザクション発行元のＣＰＵキャッシュの次ステートはＭとなる。このとき、ブリッジキャッシュの次ステートもＭとなるので（表６参照）、この場合も制約１を満足する。

【０１２２】このように、表７の規則にしたがってスプリット応答時のスヌープ出力を生成することにより、スプリットを行った場合においても前記制約１を満足することができ、この結果ＭＬＩを満たし、キャッシュ・コンシステンシを維持することができる。

【０１２３】［１．３．２］制約２を採用した場合

本実施形態において制約２を採用した場合には、バスブリッジは、スプリットを行った場合に、下位バスに発行したトランザクションの種類に基づき、表８の規則に従って上位バスに発行するスヌープ出力を決定する。

【０１２４】

【表８】

	下位バスのスヌープ結果		
下位バスに発行したトランザクション	ミス	ＨＩＴ	ＨＩＴＭ
Read Invalidate Read&Invalidate	ミス（Ｅ） ミス（Ｍ） ミス（Ｍ）	ＨＩＴ（Ｓ） ミス（Ｍ） ミス（Ｍ）	ＨＩＴ（Ｓ） ミス（Ｍ） ミス（Ｍ）

（注）括弧（）はトランザクション発行元のＣＰＵキャッシュの最終ステート

表８の規則と前記表７に示した制約１の場合の規則との相違は、バスブリッジから下位バスに発行したトランザクションがＲｅａｄで下位バスのスヌープ結果がミスの場合のスヌープ出力の決め方である。すなわち、表８の

規則では、このような場合のバスブリッジから上位バスへのスヌープ出力をミスとしている。

【０１２５】バスブリッジから下位バスに発行したトランザクションがＲｅａｄで下位バスのスヌープ結果がミスの場合、ブリッジキャッシュの次ステートはＥとなるが、制約２ではブリッジキャッシュのステートがＥの場合、その上位のＣＰＵキャッシュのステートがＥとなることを許す。したがって、この場合におけるバスブリッジからのスプリット応答トランザクションにおいてスヌープ出力をミスとすることにより、このスヌープ出力を受け取ったトランザクション発行元のＣＰＵのＣＰＵキャッシュの次ステートをＥとすることができる。

【０１２６】このように、表８にしたがってスプリット応答時のスヌープ出力を生成することにより、スプリットを行った場合においても前記制約２を満足することができ、この結果ＭＬＩを満たし、キャッシュ・コンシステンシを維持することができる。

【０１２７】［２］下位バス上に発行されたトランザクションに対する基本動作

他のバスブリッジから下位バス上に発行されたトランザクションを受け取ったバスブリッジの一連の動作について説明する。すなわち、ここでは、他のバスブリッジが自分の上位のＣＰＵからのトランザクションをスプリットして下位バスにトランザクションを発行した場合に、その下位バス・トランザクションを受け取ったバスブリッジの動作を説明する。

【０１２８】［２．１］上位バス出力判定

下位バス・トランザクションを受け取ったバスブリッジは、その下位バス・トランザクションの種類と現在のブリッジキャッシュのステートとに基づき、上位バスに発行すべきトランザクションを決定する。

【０１２９】［２．１．１］制約１を採用する場合

制約１を採用する場合には、下位バス・トランザクションを受け取ったバスブリッジは、次の表９に示す規則にしたがって、上位バスに発行するトランザクションを決定する。

【０１３０】

【表９】

	下位バス・トランザクション		
ブリッジキャッシュステート	Read	Read&Invalidate Invalidate	WriteBack
M	Read	Invalidate	×
E	○	Invalidate	×
S	○	Invalidate	×
I	○	○	○

○：上位バスにトランザクションを発行しない
×：あり得ない

すなわち、下位バス・トランザクションがＲｅａｄの場合、そのトランザクションを受け取ったバスブリッジは、自分のブリッジキャッシュのステートがＭの場合にのみ上位バスに対してＲｅａｄを発行する。ブリッジキャッシュのステートがＭの場合、そのバスブリッジの上位のＣＰＵが当該ブロックの最新データを持っている可能性があるからである。このようにしてバスブリッジから上位バスにＲｅａｄが発行された場合、そのＲｅａｄを受け取ったＣＰＵがそのＲｅａｄの要求するキャッシュブロックをＭステートで持っていれば、そのＣＰＵは上位バスに対してスヌープ出力としてＨＩＴＭをアサートすると共に、当該Ｍステートのブロックをバスブリッジに返す。このとき、そのＣＰＵのキャッシュのステートはＳに変化する。これを受け取ったバスブリッジは、このブロックを下位バスを介して下位バス・トランザクション発行元のバスブリッジに返す。そして、この下位バス・トランザクション発行元のバスブリッジは、下位バスから受け取った最新のデータをスプリット応答としてＲｅａｄの発行元のＣＰＵに返す。

【０１３１】下位バス・トランザクションがＲｅａｄの場合においてブリッジキャッシュのステートがＭ以外の場合は、ここではＣＰＵキャッシュとブリッジキャッシュは制約１を満足しているので、当該ブリッジキャッシュの上位のＣＰＵキャッシュには要求ブロックをステートＭで持っているものはない。したがって、バスブリッジは上位バスに対してトランザクションを発行する必要がない。

【０１３２】また、下位バス・トランザクションがＩｎｖａｌｉｄａｔｅまたはＲｅａｄ＆Ｉｎｖａｌｉｄａｔｅの場合、そのトランザクションを受け取ったバスブリッジは、自分のブリッジキャッシュのステートがＭ，Ｅ，Ｓの場合に上位バスに対してＩｎｖａｌｉｄａｔｅを発行する。すなわち、ブリッジキャッシュのステートがＭ，Ｅ，Ｓの場合、その上位のＣＰＵキャッシュが無効化すべきデータを持っている可能性があるので、バスブリッジは上位バスにＩｎｖａｌｉｄａｔｅを発行する。一方、ブリッジキャッシュのステートがＩの場合は、その上位のＣＰＵキャッシュにはトランザクションの対象となるブロックがないので、上位バスにトランザクションを発行する必要がない。

【０１３３】そして、下位バス・トランザクションがＷｒｉｔｅＢａｃｋの場合は、バスブリッジは、上位バスにトランザクションを発行しない。バスブリッジが下位バスにＷｒｉｔｅＢａｃｋを発行するのは、自分のブリッジキャッシュのブロックをエビクションしてメインメモリに書き戻す場合であり、ＷｒｉｔｅＢａｃｋの発行

元は他のバスブリッジに対してデータの要求も無効化の要求もしていないからである。

【０１３４】以上説明した規則にしたがってバスブリッジが上位バスに発行するトランザクションを決定することにより、ブリッジキャッシュとＣＰＵキャッシュとの制約１（表１）を満足させ、キャッシュ・コンシステンシを維持することができる。

【０１３５】［２．１．２］制約２を採用する場合

制約２を採用する場合、バスブリッジは、下位バスのトランザクションを受け取ると、そのトランザクションの種類とそのトランザクションを受け取ったときのブリッジキャッシュのステートとに基づき、表１０に示す規則にしたがって上位バスに発行するトランザクションを決定する。

【０１３６】

【表１０】

	下位バス・トランザクション		
ブリッジキャッシュステート	Read	Read&Invalidate Invalidate	WriteBack
M	Read	Invalidate	×
E	Read	Invalidate	×
S	○	Invalidate	×
I	○	○	○

○：上位バスにトランザクションを発行しない
×：あり得ない

表１０の規則は、下位バスから受け取ったトランザクションがＲｅａｄでかつブリッジキャッシュのステートがＥである場合において上位バスへＲｅａｄを発行する点が、制約１における表９の規則と異なる。

【０１３７】制約２では、ブリッジキャッシュのステートがＥのときに、その上位のＣＰＵキャッシュがステートＭとなっている（すなわち、ＣＰＵキャッシュが最新のデータを持っている）可能性がある。したがって、下位バスから受け取ったトランザクションがＲｅａｄでかつブリッジキャッシュのステートがＥである場合には、バスブリッジは上位バスに対してＲｅａｄを発行する。もしそのバスブリッジの上位のＣＰＵキャッシュの一つがＲｅａｄが要求するデータをステートＭで持っていれば、そのＣＰＵは上位バスに対してスヌープ出力としてＨＩＴＭをアサートすると共に、当該Ｍステートのブロックをバスブリッジに返す。この結果、バスブリッジは最新のデータを有することとなる。そして、バスブリッジは、最初に受け取った下位バス・トランザクションに対する応答として、この最新データを下位バスに出力する。

【０１３８】なお、表１０におけるこれ以外の場合についての規則は、表９の規則と同様である。

【０１３９】以上説明した表１０の規則にしたがってバスブリッジが上位バスに発行するトランザクションを決定することにより、表２に示したブリッジキャッシュとＣＰＵキャッシュとの制約２を満足させ、キャッシュ・コンシステンシを維持することができる。

【０１４０】なお、上記表９又は表１０の規則にしたがってバスブリッジから上位バスにトランザクションが発行された場合、その上位バスに接続された各ＣＰＵは、階層を意識せず、そのトランザクションが他のＣＰＵから発行された場合と同様に応答する。したがって、例えばバスブリッジから上位バスにＩｎｖａｌｉｄａｔｅが発行されたときに、その上位バス上にＩｎｖａｌｉｄａｔｅの対象ブロックをステートＭで持っているＣＰＵがあった場合は、そのＣＰＵは上位バスに対してＨＩＴＭをアサートすると同時にその対象ブロックのデータを上位バスに出力する。

【０１４１】［２．１．３］変形例

［２．１．１］及び［２．１．２］で説明した上位バス出力判定方式によれば、下位バス・トランザクションがＩｎｖａｌｉｄａｔｅの場合、これを受け取ったバスブリッジのブリッジキャッシュのステートがＭであれば、当該バスブリッジは、原則として上位バスに対してＩｎｖａｌｉｄａｔｅを発行する（表９及び表１０参照）。

【０１４２】ところが、あるＣＰＵキャッシュにおいてエビクションが発生し、ステートＭのブロックがそのＣＰＵの下位のブリッジキャッシュに書き戻された場合、そのブリッジキャッシュのステートはＭとなるのに対し、元のＣＰＵキャッシュと同じ上位バスに接続されたＣＰＵキャッシュはすべてステートＩとなる。したがって、この場合、バスブリッジは、ＣＰＵから書き戻されたブロックに対するＩｎｖａｌｉｄａｔｅを下位バスから受け取ったとしても、上位バスに対してＩｎｖａｌｉｄａｔｅを発行する必要はない。表９又は表１０に示した方式においては、このような場合でも上位バスに対するＩｎｖａｌｉｄａｔｅを発行することとなり、不要なＩｎｖａｌｉｄａｔｅの発行を行っていることとなる。

【０１４３】また、［２．１．１］及び［２．１．２］で説明した上位バス出力判定方式によれば、下位バス・トランザクションがＲｅａｄの場合、これを受け取ったバスブリッジのブリッジキャッシュのステートがＭであれば、当該バスブリッジは、原則として上位バスに対してＲｅａｄを発行する（表９及び表１０参照）。

【０１４４】ところが、あるＣＰＵがステートＭのブロックを持っているときに、そのＣＰＵと同一上位バス上のＣＰＵがそのブロックをＲｅａｄした場合、当該上位バス上のＣＰＵキャッシュのステートはＳ又はＩとな

り、それらＣＰＵキャッシュの下位のブリッジキャッシュのステートはＭとなる。すなわち、当該ブロックの最新のデータはブリッジキャッシュが所有していることとなる。この場合、バスブリッジは、当該ブロックに対するＲｅａｄを下位バスから受け取ったとしても、上位バスに対してＲｅａｄを発行する必要はない。表９又は表１０に示した方式においては、このような場合でも上位バスに対するＲｅａｄを発行することとなり、不要なＲｅａｄの発行を行っていることとなる。

【０１４５】この変形例は、以上に説明したような不要なトランザクションの発行を抑制することを目的とする。

【０１４６】このため、本変形例では、バスブリッジにおいて、ブリッジキャッシュの各ブロックごとに状態フラグを設け、この状態フラグによってトランザクションの発行を制御する。

【０１４７】図２は、この変形例の概略構成を示す図である。図２において、上位バス２０と下位バス４０との間でトランザクションの制御を行うバスブリッジ３０は、バスブリッジ制御部６０と上位キャッシュ状態フラグ記憶部８０とを含む。上位キャッシュ状態フラグ記憶部８０は、ブリッジキャッシュ３１の各ブロックごとに状態フラグを有する。この状態フラグは２ビットであり、各ビット（Ａ）、（Ｂ）は、それぞれ次の状態を示すためのビットである。

【０１４８】（Ａ）そのフラグに対応するブリッジキャッシュのブロックをＩｎｖａｌｉｄａｔｅするために、上位バス２０上の各ＣＰＵ１０及び１１のキャッシュをＩｎｖａｌｉｄａｔｅする必要がない。すなわち、ＭＥＳＩプロトコルでは、上位バス２０上の各ＣＰＵのキャッシュのステートがすべてＩである

（Ｂ）そのフラグに対応するブリッジキャッシュ３１のブロックのデータを下位バス４０に出力するために、上位バス２０に対してＲｅａｄを発行する必要がない。すなわち、上位バス２０上の各ＣＰＵのキャッシュはＩ又はＳである

バスブリッジ制御部６０は、上位バス２０・下位バス４０間のトランザクションのやり取り及びブリッジキャッシュ３１の制御のほかに、上位キャッシュ状態フラグ記憶部８０の制御を行う。

【０１４９】バスブリッジ制御部６０は、ＣＰＵ１０又は１１から上位バス２０上に発行されたトランザクションをモニタする。そして、そのトランザクションの種類に基づき、上位キャッシュ状態フラグ記憶部８０の当該ブロックに対する状態フラグを制御する。すなわち、バスブリッジ制御部６０は、上位バス２０上にＷｒｉｔｅＢａｃｋが発行されたのを検出したときには、上位キャッシュ状態フラグ制御部８０における該当ブロックに対する状態フラグのビット（Ａ）をセットする。また、バスブリッジ制御部６０は、上位バス２０上にＲｅａｄが発行されたのを検出したときには、上位キャッシュ状態フラグ制御部８０における該当ブロックに対する状態フラグのビット（Ｂ）をセットする。そして、上位バス２０上にＩｎｖａｌｉｄａｔｅ系トランザクションが発行された場合には、バスブリッジ制御部６０は、該当ブロックに対する状態フラグを両ビットともリセットする（ＣＰＵがＩｎｖａｌｉｄａｔｅ系を発行するのは、該当ブロックに対して書き込みを行うときだからである）。

【０１５０】そして、バスブリッジ制御部６０は、下位バス４０からＩｎｖａｌｉｄａｔｅを受け取った場合、まずブリッジキャッシュ３１をスヌープしてＩｎｖａｌｉｄａｔｅ対象ブロックのステートを調べる。このステートがＭの場合は、上位キャッシュ状態フラグ記憶部８０を検索して、当該ブロックの状態フラグのビット（Ａ）を調べる。そして、このビット（Ａ）がリセット状態である場合にのみ、バスブリッジ制御部６０は上位バス２０に対してＩｎｖａｌｉｄａｔｅを発行する。ビット（Ａ）がセット状態の場合は、上位バス２０上のＣＰＵキャッシュはすべてステートＩなのでＩｎｖａｌｉｄａｔｅの発行は行わない。

【０１５１】また、バスブリッジ制御部６０は、下位バス４０からＲｅａｄを受け取った場合、ブリッジキャッシュ３１をスヌープしてＲｅａｄ対象のブロックのステートを調べる。このステートがＭの場合は、さらに上位キャッシュ状態フラグ記憶部８０を検索して、当該ブロックの状態フラグのビット（Ｂ）を調べる。そして、このビット（Ｂ）がリセット状態の場合のみ、バスブリッジ制御部６０は上位バス２０に対してＲｅａｄを発行する。

【０１５２】このように、図２の構成では、バスブリッジ制御部６０において、上位バス２０上のトランザクションをモニタすることにより、上位バス２０上のＣＰＵキャッシュの状態を把握する。そして、この結果得た情報を用いて下位バス４０から上位バス２０へのトランザクションの転送を制御することにより、上位バス２０上へ発行されるトランザクションを減らすことができる。したがって、図２の構成によれば、上位バス２０の負荷を低減することができる。

【０１５３】［２．２］下位バスに対するスヌープ出力

下位バスからトランザクションを受け取ったバスブリッジが、表９又は表１０の規則にしたがって上位バスにトランザクションを発行すると、上位バスの各ＣＰＵからそのトランザクションに対する応答が当該バスブリッジに返される。それら上位バスからの応答に基づき、バスブリッジは下位バスに対して応答を発する。この応答の時に、バスブリッジは、下位バス・トランザクション発行元のバスブリッジに対するスヌープ出力を下位バス上に発する。

【０１５４】ここでは、他のバスブリッジから下位バス

に発行されたトランザクションを受け取ったバスブリッジが、このトランザクションに対して下位バスに返すスヌープ出力の決定の方法について説明する。

【０１５５】［２．２．１］制約１を採用する場合

制約１を採用する場合、バスブリッジから下位バスへ発行するスヌープ出力は、当該バスブリッジが受け取った下位バス・トランザクションの種類と、下位バストランザクションを受け取った時点でのブリッジキャッシュのステートとに基づき、次の表１１の規則にしたがって決定する。

【０１５６】

【表１１】

	下位バス・トランザクション		
ブリッジキャッシュステート	Read	Read&Invalidate Invalidate	WriteBack
M	HITM	HITM	×
E	HIT	ミス	×
S	HIT	ミス	×
I	ミス	ミス	ミス

×：あり得ない

まず、下位バス・トランザクションがＲｅａｄであった場合について説明する。この場合においてブリッジキャッシュのステートがＩの場合には、当該ブリッジキャッシュ及びその上位のＣＰＵキャッシュのいずれもＲｅａｄが要求するデータを持っていないので、バスブリッジは下位バスに対するスヌープ出力をミスとする。また、ブリッジキャッシュがＳ又はＥの場合には、当該ブリッジキャッシュが下位バスのＲｅａｄの要求するデータを持っており、しかもそのデータの内容はメインメモリと同内容なので、バスブリッジは下位バスに対するスヌープ出力をＨＩＴとする。特に、制約１を採用する場合は、ブリッジキャッシュがＥの場合にはその上位のＣＰＵキャッシュはＭとなり得ないので、ブリッジキャッシュがＥの場合はバスブリッジは上位バスをスヌープするまでもなく、下位バスに対してＨＩＴをアサートすれば足りる。また、ブリッジキャッシュのステートがＭの場合は、ブリッジキャッシュ又はその上位のＣＰＵキャッシュのいずれかがそのＲｅａｄの要求するデータを持っているので、下位バスに対してＨＩＴＭをアサートする。なお、ブリッジキャッシュのステートがＭの場合、前述したようにバスブリッジから上位バスにＲｅａｄが発行される。このＲｅａｄに対する上位バスのスヌープ結果がＨＩＴＭだった場合は、バスブリッジは上位バスのＣＰＵキャッシュから最新のデータを得てブリッジキャッシュに格納すると共に下位バス上にその最新データを出力する。一方、Ｒｅａｄに対する上位バススヌープ結果がＨＩＴ又はミスであった場合には、バスブリッジはブリッジキャッシュが有するデータを最新データとし下位バスに出力する。

【０１５７】次に、下位バス・トランザクションがＲｅａｄ＆ＩｎｖａｌｉｄａｔｅまたはＩｎｖａｌｉｄａｔｅであった場合について説明する。この場合において、ブリッジキャッシュのステートがＩ，Ｓ又はＥであった場合には、バスブリッジは下位バスに対するスヌープ出力をミスとし、ブリッジキャッシュのステートがＭであった場合には下位バスに対するスヌープ出力をＨＩＴＭとする。Ｉｎｖａｌｉｄａｔｅ系トランザクションは、対象とするキャッシュブロックを無効化（すなわちＩステートに変える）するトランザクションであるから、ブリッジキャッシュの当該ブロックの次ステートはＩとなるので、下位バスに対するスヌープ出力もミスとすればよい。ただし、ブリッジキャッシュのステートがＭであった場合には、そのブリッジキャッシュ（又は上位のＣＰＵキャッシュ）の持つ最新のデータをＲｅａｄ発行元のバスブリッジに返しかつメインメモリにも書き戻す必要があるため、下位バスにはＨＩＴＭをアサートする。

【０１５８】あるバスブリッジから下位バスに発行されたトランザクションがＷｒｉｔｅＢａｃｋであった場合には、他のバスブリッジのキャッシュのステートはＩ以外にはないので、他のバスブリッジはスヌープ出力をミスとする。

【０１５９】なお、表１１に示した規則において特徴的なのは、バスブリッジから下位バスへのスヌープ出力を決定する際に、上位バスのＣＰＵの情報が不要なことである。したがって、バスブリッジは、下位バスからトランザクションを受け取ると、そのトランザクションの種類とそのときのブリッジキャッシュとステートを調べるだけで、下位バスに対するスヌープ出力を生成できる。これは、制約１のごとくＣＰＵキャッシュのステートを制限することのメリットの一つである。

【０１６０】以上説明した規則に従ってバスブリッジから下位バスに対するスヌープ出力を生成することにより、キャッシュ・コンシステンシを維持することが可能となる。

【０１６１】［２．２．２］制約２を採用する場合

制約２を採用した場合、バスブリッジから下位バスへ発行するスヌープ出力は、当該バスブリッジが受け取った下位バス・トランザクションの種類と、そのトランザクションを受け取った時点でのブリッジキャッシュのステートと、上位バスのスヌープ結果とに基づき、次の表１２の規則にしたがって決定する。

【０１６２】

【表１２】

	下位バス・トランザクション				
ブリッジキャッシュステート	Read		Read&Invalidate Invalidate		WriteBack
	上位バススヌープ結果 HITM	上位バススヌープ結果 HITM以外	上位バススヌープ結果 HITM	上位バススヌープ結果 HITM以外	
M	HITM	HITM	HITM	HITM	×
E	HITM	HIT	HITM	ミス	×
S	×	HIT	×	ミス	×
I	×	ミス	×	ミス	ミス

×：あり得ない

表１２の規則は、表１１に示した制約１の場合の規則と異なり、上位バスのスヌープ結果によって下位バスへのスヌープ出力を区別している。ここでいう上位バスのスヌープ結果とは、下位バス・トランザクションについてバスブリッジが表１０の規則にしたがって生成した上位バス・トランザクションに対する上位バスのスヌープ結果のことである。

【０１６３】表１２の規則において、下位バス・トランザクションの種類がＷｒｉｔｅＢａｃｋの場合、及び下位バス・トランザクションの種類がＲｅａｄ、Ｉｎｖａｌｉｄａｔｅ又はＲｅａｄ＆Ｉｎｖａｌｉｄａｔｅの場合において上位バスのスプリット結果がＨＩＴＭ以外の場合のスヌープ出力の決定の規則は、表１１の場合と同様である。

【０１６４】表１２の規則の特徴点は、下位バス・トランザクションの種類がＲｅａｄ、Ｉｎｖａｌｉｄａｔｅ又はＲｅａｄ＆ＩｎｖａｌｉｄａｔｅでブリッジキャッシュのステートがＥの場合において、上位バスのスヌープ結果がＨＩＴＭとなると、下位バスに対するスヌープ出力をＨＩＴＭとする点である。

【０１６５】これは、制約２において、ブリッジキャッシュのステートがＥの場合にはその上位のＣＰＵキャッシュがステートＭとなり得るためである。すなわち、ブリッジキャッシュのステートがＥのときに、バスブリッジから上位バスに発行したトランザクションに対するスヌープ結果がＨＩＴＭであった場合には、上位バス上のＣＰＵがステートＭのデータを持っていたことになり、この結果を下位バス・トランザクションの発行元のバスブリッジへ知らせるために、下位バスへのスヌープ出力をＨＩＴＭとする。

【０１６６】以上説明した規則に従ってバスブリッジから下位バスに対するスヌープ出力を生成することにより、キャッシュ・コンシステンシを維持することが可能となる。

【０１６７】［２．３］ブリッジキャッシュの次ステート

次に、下位バスに発行されたトランザクションを受け取ったバスブリッジのブリッジキャッシュのステートの遷移について説明する。なお、ブリッジキャッシュの次ステートの決定の仕方は、制約１及び制約２のいずれを採用した場合も同じである。

【０１６８】まず、下位バス・トランザクションを受け取ったバスブリッジが上位バスに対してトランザクションを発行しない場合について説明する。

【０１６９】下位バス・トランザクションがＲｅａｄであった場合において、それを受け取ったバスブリッジのブリッジキャッシュのステートがＩであった場合には、そのブリッジキャッシュの次ステートはＩとなる。また、Ｒｅａｄを受け取ったブリッジキャッシュのステートがＳ又はＥであった場合には、そのブリッジキャッシュの次ステートはＳとなる。

【０１７０】下位バス・トランザクションがＩｎｖａｌｉｄａｔｅ系であった場合には、それを受け取ったバスブリッジのブリッジキャッシュの次ステートはＩとなる。

【０１７１】下位バス・トランザクションがＷｒｉｔｅＢａｃｋであった場合には、それを受け取ったバスブリッジのブリッジキャッシュのステートは変化しない。

【０１７２】次に、下位バス・トランザクションを受け取ったバスブリッジが上位バスに対してトランザクションを発行する場合について説明する。この場合、ブリッジキャッシュの次ステートは、表１３に示す規則にしたがって決定する。

【０１７３】

【表１３】

	上位バスのスヌープ結果		
上位バスに発行したトランザクション	ミス	ＨＩＴ	ＨＩＴＭ
Read	S	S	S
Invalidate	I	×	I
Read&Invalidate	I	×	I

×：あり得ない

表１３は、バスブリッジが上位バスに発行したトランザクションとそのトランザクションに対する上位バスのスヌープ結果との各組合わせに対するブリッジキャッシュの次ステートが示されている。

【０１７４】この規則では、バスブリッジが上位バスにＲｅａｄを発行した場合には、上位バスのスヌープ結果にかかわらず、ブリッジキャッシュの次ステートをＳとする。一方、バスブリッジが上位バスにＩｎｖａｌｉｄａｔｅ系トランザクションを発行した場合には、ブリッジキャッシュの次ステートをＩとする。

【０１７５】このような規則に従えば、上位バスのスヌープ結果を用いずに容易にブリッジキャッシュの次ステートを決定することができる。

【０１７６】下位バス・トランザクションを受け取ったバスブリッジが上位バスに対してトランザクションを発行した場合における、ブリッジキャッシュの次ステートの決め方にはもう一つの方法がある。その決め方の規則を表１４に示す。

【０１７７】

【表１４】

	上位バスのスヌープ結果		
上位バスに発行したトランザクション	ミス	ＨＩＴ	ＨＩＴＭ
Read	I	S	S
Invalidate	I	×	I
Read&Invalidate	I	×	I

×：あり得ない

この規則が表１３の規則と異なる点は、上位バスにＲｅａｄを発行した場合において、上位バスのスヌープ結果に基づいてブリッジキャッシュの次ステートを決定している点である。表１４の規則では、上位バスにＲｅａｄを発行した場合、上位バスのスヌープ結果がＨＩＴ又はＨＩＴＭの場合には表１３の規則と同様ブリッジキャッシュの次ステートをＳとするが、上位バスのスヌープ結果がミスの場合はブリッジキャッシュの次ステートをＩとする。このようにすることによって次のような効果が得られる。

【０１７８】すなわち、上位バスのスヌープ結果がミスということは、当該バスブリッジの上位バス上のＣＰＵキャッシュが現在すべてステートＩであるということを意味する。従って、このような場合にブリッジキャッシュの次ステートをＩとしても、キャッシュ・コンシステンシは満足される。また、上位バス上のＣＰＵキャッシュがすべてＩということは、上位バス上のＣＰＵはＲｅａｄトランザクションが要求するブロックを現在全く使用していないということを意味するので、このような場合にブリッジキャッシュのみが当該ブロックを持っていたとしても無駄になる可能性が高い。そこで、表１４に示したように上位バスのスヌープ結果がミスのときにブリッジキャッシュの次ステートをＩとすることにより、ブリッジキャッシュに空きブロックが生まれ、キャッシュの有効利用を図ることができる。

【０１７９】以上、上位バスに発行されたトランザクションを受け取ったときのバスブリッジの基本動作と、下位バスに発行されたトランザクションを受け取ったときのバスブリッジの基本動作とを説明した。次は、ブリッジキャッシュのエビクション時のバスブリッジの動作、及び上位バスのトランザクションに対するリトライについて説明する。

【０１８０】［３］ブリッジキャッシュのエビクションを行う場合のバスブリッジの動作

キャッシュメモリには様々な方式があるが、制御の容易さやコストなどからダイレクトマッピング方式やセットアソシアティブ方式が広く用いられている。これらの方式においては、一つのキャッシュブロックに対して複数のメモリアドレスを対応させているため、あるデータを読み込もうとする場合に、そのデータに対応するキャッシュメモリのブロックがそのブロックに対応する他のアドレスのデータによって使用され、空きがない場合がある。このような場合に、現在使用されているブロックを無効化して空きブロックを作る処理のことをエビクションと呼ぶ。キャッシュに読み込もうとしていたデータは、エビクションの結果できた空きブロックに格納される。

【０１８１】さて、ブリッジキャッシュにてエビクションを行った場合、上位バス上には当該エビクションを行ったブロックを持っているＣＰＵキャッシュが存在する可能性がある。従って、ブリッジキャッシュのエビクションを行った場合には、上位バスのＣＰＵキャッシュの

当該ブロックを無効化する必要がある場合がある。また、エビクション対象のブロックが最新のデータであった場合、そのデータをメインメモリに書き戻す必要がある。

【０１８２】本実施形態において、上位バスのトランザクションを検出したバスブリッジは、ブリッジキャッシュを検索して当該トランザクションのアドレスに対応するブロックが空いているか否かを調べ、空きブロックがない場合にエビクションを行う。エビクションにおいては、ブリッジキャッシュの上位のＣＰＵキャッシュの該当ブロックを無効化し、ブリッジキャッシュの該当ブロックのデータをメインメモリへ書き戻したのち、ブリッジキャッシュの該当ブロックをステートをＩに変更して使用可能な状態とする。

【０１８３】［３．１］上位バス上のＣＰＵキャッシュの無効化

上位のＣＰＵキャッシュ内のエビクション対象ブロックを無効化するためには、バスブリッジは上位バスに対してＩｎｖａｌｉｄａｔｅを発行する。このＩｎｖａｌｉｄａｔｅの発行は、次の表１５に基づいて行う。

【０１８４】

【表１５】

ブリッジキャッシュステート	上位バスに発行するトランザクション
M	Invalidate
E	Invalidate
S	Invalidate
I	O

Ｏ：発行しない

すなわち、本実施形態では、バスブリッジは、自らのブリッジキャッシュにおけるエビクション対象のブロックのステートに基づき、上位バスにＩｎｖａｌｉｄａｔｅを発行するか否かを決定する。なお、この表１５及び次に説明する表１６及び１７では、ブリッジキャッシュのステートの意味が、これまでの表３や４などの場合と異なる。表３等において、ブリッジキャッシュのステートといった場合、トランザクションに対するステート、すなわちトランザクションの要求データがブリッジキャッシュに有るか無いか等を意味した。これに対して、表１５等では、ブリッジキャッシュのステートとは、トランザクションとは無関係に、ブロック自体に格納されているデータのステートを意味する。

【０１８５】表１５の規則に従う場合、バスブリッジは、ブリッジキャッシュのステートがＭ，Ｅ，Ｓの場合に上位バスにＩｎｖａｌｉｄａｔｅを発行する。ブリッジキャッシュのステートがＭ，Ｅ，Ｓの場合、上位のＣＰＵキャッシュが当該ブロックを持っている可能性があるからである。

【０１８６】なお、このＩｎｖａｌｉｄａｔｅを受け取った上位バスの各ＣＰＵは、自らのＣＰＵキャッシュを検索してＩｎｖａｌｉｄａｔｅ対象のブロックの状態を調べ、上位バスに対してスヌープ出力を返す。またこのとき、上位バス上のＣＰＵのなかに当該ブロックをステートＭで所有しているものがあれば、そのＣＰＵは当該ブロックのデータを上位バスに出力する。バスブリッジは、ＣＰＵから上位バス上に出力されたデータを受け取り、そのデータにより当該エビクション対象のブロックを更新する。

【０１８７】なお、表１５の規則は、制約１及び制約２に共通のものである。

【０１８８】［３．２］ブリッジキャッシュのデータのメインメモリへの書き戻し

また、ブリッジキャッシュ内の最新データをメインメモリに対して書き戻すためには、バスブリッジは下位バスにＷｒｉｔｅＢａｃｋを発行する。

【０１８９】［３．２．１］制約１を採用する場合

制約１を採用する場合は、バスブリッジから下位バスへのＷｒｉｔｅＢａｃｋの発行は、表１６の規則に基づいて行う。

【０１９０】

【表１６】

ブリッジキャッシュステート	下位バスに発行するトランザクション
M	WriteBack
E	O
S	O
I	O

Ｏ：発行しない

バスブリッジは、ブリッジキャッシュのステートに基づいて、下位バスにＷｒｉｔｅＢａｃｋを発行するか否かを決定する。表１６に示すように、バスブリッジは、ブリッジキャッシュがステートＭのときにのみ、下位バスに対してＷｒｉｔｅＢａｃｋを発行する。ブリッジキャッシュがステートＭ以外の場合は、そのブリッジキャッシュ自体のもっているデータはメインメモリと同内容であり、また制約１（表１）を満足していればその上位のＣＰＵキャッシュが当該ブロックの最新データを持っていることはないから、そのような場合にはデータの下記戻しは不要だからである。

【０１９１】したがって、ブリッジキャッシュがＭのときには、Ｉｎｖａｌｉｄａｔｅによって上位バス上のＣＰＵからブリッジキャッシュに書き戻された当該ブロックの最新データ、もしくはそのような上位バスからの書き戻しがなかった場合はブリッジキャッシュ自身が持っているデータが、ＷｒｉｔｅＢａｃｋトランザクションによってメインメモリに書き戻される。一方、ブリッジキャッシュがＥまたはＳのときには、バスブリッジはメインメモリへのデータの書き戻しは行わず、単に当該ブロックのステートをＩに変更し、上位バスに当該ブロッ

クに対するＩｎｖａｌｉｄａｔｅを発行する。ブリッジキャッシュのステートがＩのときは、エビクション自体が行われず、従って上位バスに対しても下位バスに対してもトランザクションは発行されない。

【０１９２】［３．２．２］制約２を採用する場合

制約２を採用する場合、ブリッジキャッシュのステートがＥのときに、そのブリッジキャッシュの上位のＣＰＵキャッシュがステートＭとなっている可能性がある。したがって、ブリッジキャッシュのステートがＥの場合には、バスブリッジは、上位バス上のＣＰＵキャッシュを調べ、もしステートＭのＣＰＵキャッシュがあれば、そのＣＰＵキャッシュからステートＭのデータを取得し、これをメインメモリに書き戻す必要がある。上位バス上のＣＰＵキャッシュがステートＭであるか否かは、前記［３．１］で説明したＩｎｖａｌｉｄａｔｅに対する上位バスからのスヌープ結果から知ることができる。すなわち、上位バス上のＣＰＵキャッシュがステートＭの場合、そのＣＰＵキャッシュは、バスブリッジからＩｎｖａｌｉｄａｔｅを受け取ると、スヌープ出力としてＨＩＴＭをアサートするので、バスブリッジの上位バス・スヌープ結果はＨＩＴＭとなる。

【０１９３】なお、ステートＭのＣＰＵキャッシュは、バスブリッジからＩｎｖａｌｉｄａｔｅを受け取ると、応答としてステートＭのデータを上位バス上に出力するので、バスブリッジは上位ＣＰＵキャッシュからのステートＭのデータを自動的に取得することができる。

【０１９４】したがって、制約２を採用する場合におけるバスブリッジから下位バスへのＷｒｉｔｅＢａｃｋの発行は、ブリッジキャッシュのステート及びＩｎｖａｌｉｄａｔｅに対する上位バスのスヌープ結果とに基づき、次の表１７の規則に従って行う。

【０１９５】

【表１７】

ブリッジキャッシュステート	下位バスに発行するトランザクション	
	上位バスのスヌープ結果HITM	上位バスのスヌープ結果HITM以外
M	WriteBack	WriteBack
E	WriteBack	○
S	○	○
I	○	○

○：発行しない

制約２を採用した場合、ブリッジキャッシュのステートがＥ又はＭのときに、上位のＣＰＵキャッシュがステートＭになる可能性がある。したがって、ブリッジキャッシュのステートがＥ又はＭの場合に、上位バス・スヌープ結果がＨＩＴＭになる可能性がある。したがって、表１７では、ブリッジキャッシュのステートがＥ又はＭで、上位バス・スヌープ結果がＨＩＴＭの場合には、下位バスに対してＷｒｉｔｅＢａｃｋを発行する。このＷｒｉｔｅＢａｃｋによってブリッジキャッシュからメインメモリに書き戻されるデータは、バスブリッジ（及びブリッジキャッシュ）が上位のＣＰＵキャッシュから取得したステートＭのデータである。

【０１９６】また、ブリッジキャッシュのステートがＭの場合であっても、上位のＣＰＵキャッシュがすべてステートＭ以外の場合もある。このような状態は、例えば、あるＣＰＵキャッシュのステートＭのデータがエビクションによりブリッジキャッシュに書き戻された場合などに起こる。この場合、Ｉｎｖａｌｉｄａｔｅに対する上位バスのスヌープ結果はＨＩＴＭ以外となる。このとき、最新のデータはブリッジキャッシュが所有していることになる。したがって、表１７において、ブリッジキャッシュのステートがＭで、上位バス・スヌープ結果がＨＩＴＭ以外の場合には、バスブリッジは下位バスに対してＷｒｉｔｅＢａｃｋを発行する。このとき、ブリッジキャッシュからメインメモリに書き戻されるデータは、ブリッジキャッシュが所有しているデータである。

【０１９７】なお、ブリッジキャッシュがステートＥで、上位バス・スヌープ結果がＨＩＴＭ以外の場合には、上位のＣＰＵキャッシュがステートＭである可能性はないので、この場合にはＷｒｉｔｅＢａｃｋは発行しない。

【０１９８】以上説明した規則にしたがって上位バスへのＩｎｖａｌｉｄａｔｅの発行及び下位バスへのＷｒｉｔｅＢａｃｋの発行を行うことにより、システム全体のキャッシュ・コンシステンシを維持しつつブリッジキャッシュのエビクションを行うことができる。

【０１９９】［３．３］トランザクション処理とエビクション処理との関係

次に、本実施形態におけるブリッジキャッシュのエビクションのためのバスブリッジの処理動作と、そのエビクションの起因となった上位バス・トランザクションのためのバスブリッジの処理動作との関係について説明する。

【０２００】前述したように、上位バス・トランザクションのアドレスに対応するブリッジキャッシュのブロックが空きブロックにならない限り、バスブリッジはその

上位バス・トランザクションの処理を進めることはできない。

【０２０１】（ａ）そこで、まず考えられるのは、エビクションを先に行って空きブロックを確保した後、上位バス・トランザクションについての処理を行うという方式である。

【０２０２】エビクション処理においては、上位ＣＰＵキャッシュの無効化のためバスブリッジからＩｎｖａｌｉｄａｔｅを発行する必要があるので、この時点で上位バスを使用しているトランザクション発行元ＣＰＵに上位バスをいったん解放させる必要がある。このために、本実施形態では、ブリッジキャッシュのエビクションが必要な場合には、トランザクション発行元のＣＰＵにトランザクションのリトライを指示して、いったん上位バスを解放させる。リトライとは、現在発行中のトランザクションをいったん終了してバスを解放した後、所定時間経過後に再び同じトランザクションを発行する処理である。なお、この場合、ＣＰＵはリトライ機能を有するものを用いる。

【０２０３】このときの処理の流れは以下のようになる。まず、上位バス・トランザクションを受け取ったバスブリッジは、ブリッジキャッシュをスヌープしてエビクションが必要か否かを判定する。この結果、エビクションの必要があると判定した場合は、上位バスに対してリトライ終了信号を出力してトランザクション発行元ＣＰＵにリトライを指示し、エビクション処理（上位ＣＰＵのＩｎｖａｌｉｄａｔｅ及びメインメモリへのＷｒｉｔｅＢａｃｋ）を行う。一方、バスブリッジからのリトライ終了信号を受け取ったトランザクション発行元ＣＰＵは、いったんトランザクションを終了して上位バスを解放し、その後所定時間経過後に同じトランザクションを再発行する。この再発行時には、バスブリッジにおけるエビクション処理が終了しており、バスブリッジは当該トランザクションの処理を進めることができる。

【０２０４】（ｂ）以上説明したようなリトライを行う処理では、ＣＰＵが最初にトランザクションを発行してから、最終的に当該トランザクションが完結するまでの時間が長くなってしまう。また、ブリッジキャッシュのエビクションが完了した後ＣＰＵからトランザクションが再発行されるまでの間に、エビクションにより設けた空きブロックが他のＣＰＵからのトランザクションにより先に使用されてしまう可能性もあり、このような場合は再びエビクションを行わなければならなくなる場合がある。

【０２０５】このような場合に対処するための方法としては、バスブリッジ内に特別のバッファを設け、エビクション対象のブロックをそのバッファに一時的に退避してブリッジキャッシュを空け、先にＣＰＵからのトランザクションの処理を行ったのちにそのバッファに退避したブロックのデータを用いてエビクション処理を行うという方式がある。

【０２０６】図３は、この方式におけるバスブリッジの構成を説明するための概略構成図である。図３において、上位バス２０と下位バス４０との間のトランザクションの制御を行うバスブリッジ３０は、これまで説明したバスブリッジ基本動作を実行するバスブリッジ制御部６０と、エビクション対象のキャッシュブロックを一時的に格納するエビクションバッファ６２とを含む。エビクションバッファ６２は、退避ブロックのデータ内容を格納するデータバッファ部６６と、退避ブロックのアドレスを格納するアドレスレジスタ部６４とを有する。なお、エビクションバッファ６２は、数ブロック分のデータ及びアドレスを記憶することができ、複数のトランザクションの並行処理に対応可能となっている。エビクションバッファ６２は必要な容量が小さいのでサイズも小さく、バスブリッジ制御部６０と同一のチップの上に作り込むことができる（ちなみに、ブリッジキャッシュ３１はＭＬＩを満たすためにサイズが大きくなるため、バスブリッジ制御部６０と同一のチップ上に形成することは困難である）。

【０２０７】図３において、上位バス２０に発行されたトランザクションをバスブリッジ３０が受け取った場合、バスブリッジ制御部６０は、前述のようにブリッジキャッシュ３１をスヌープしてトランザクションをスプリットすべきか否かなどの判定を行う。このとき、バスブリッジ制御部６０は、トランザクションのアドレスに対応するブリッジキャッシュ３１内のブロックの使用状況を検出し、検出の結果当該ブロックが空いていないことが分かると、当該ブロックのデータをブリッジキャッシュ３１から読み出してエビクションバッファ６２のデータバッファ部６６に格納し、当該ブロックのアドレスをアドレスレジスタに登録する。これ同時に、バスブリッジ制御部６０は、ブリッジキャッシュ３１の当該ブロックをステートＩに変更して使用可能な状態とする。

【０２０８】このような処理の結果、上位バス・トランザクションの処理が可能となるので、バスブリッジ制御部６０は、前記［１］及び［２］節で説明した一連の処理を行って、そのトランザクションを完結させる。すなわち、バスブリッジ制御部６０は、トランザクションのスプリットを行って下位バス４０にトランザクションを発行し、この結果下位バス４０の他のエージェント（メインメモリ又はバスブリッジ）から得たデータを、スプリット応答トランザクションとして上位バス２０に出力する。

【０２０９】そして、この後、バスブリッジ制御部６０は、エビクションバッファ６２に一時退避したブロックに対するエビクション処理を行う。すなわち、バスブリッジ制御部６０は、退避ブロックに対するＩｎｖａｌｉｄａｔｅトランザクションを上位バス２０に発行してＣＰＵキャッシュとブリッジキャッシュとのコンシステン

シの維持を図り、当該退避ブロックがMステートである場合には下位バス４０にＷｒｉｔｅＢａｃｋトランザクションを発行し、メインメモリに当該退避ブロックを書き戻す。なお、この場合において、バスブリッジ制御部６０は、退避ブロックに対するＩｎｖａｌｉｄａｔｅに対して、上位バス上のＣＰＵからMステートのブロックが書き戻されてきた場合は、そのブロックのアドレスを判別して、この書き戻されてきたブロックをブリッジキャッシュ３１ではなくエビクションバッファ６２に書き込む。この場合、上位バスから書き戻されてきたデータは、最初にブリッジキャッシュ３１から退避したデータの上に上書きされる。したがって、エビクションバッファ６２のブロックは、メインメモリに書き戻すときには、必ず最新のデータとなっている。

【０２１０】このように、図３の構成によれば、トランザクションに対する処理をエビクションに先行して行うことができるため、ＣＰＵが最初にトランザクションを発行してからそのトランザクションが完結するまでの時間が大幅に短縮される。

【０２１１】なお、以上の説明では、図３の構成における実質的なエビクションの処理はトランザクションの処理が完結した後に行うとしていたが、この順序は必須のものではない。例えば、バスにおいてパイプライン処理が可能である場合には、上位バスが使用されていなければＩｎｖａｌｉｄａｔｅを発行することができ、また上位バスのＩｎｖａｌｉｄａｔｅが完了すればいつでも下位バスにＷｒｉｔｅＢａｃｋを発行することができる。したがって、この場合には、トランザクションに対する処理が完結する前にＩｎｖａｌｉｄａｔｅやＷｒｉｔｅＢａｃｋを発行することも可能である。

【０２１２】また、図３の構成では、エビクション対象のブロックの退避場所としてエビクションバッファ６２を用いたが、ブロック退避場所としては、図４に示すような補助キャッシュ６８を用いることもできる。図４において、補助キャッシュ６８は、数ブロックを格納することができる小容量のフルアソシアティブ・キャッシュである。図３のエビクションバッファ６２と図４の補助キャッシュ６８との違いは、エビクションバッファ６２に格納されたブロックはバスブリッジ制御部６０から読みだすことはできないが、補助キャッシュ６８に格納されたブロックはバスブリッジ制御部６０から読み出すことができるという点である。したがって、バスブリッジ制御部６０が退避中のブロックを要求するトランザクションを上位バスから受け取った場合、図３の構成ではバスブリッジ制御部６０はそのトランザクションに対する応答を行うことができないが、図４の構成によれば、補助キャッシュ６８から当該ブロックを読み出して応答することができる。ただし、補助キャッシュ６８の制御はエビクションバッファ６２の制御よりも複雑となるため、図４のバスブリッジ制御部６０の回路構成は、図３のそれよりも複雑かつ大規模になる。

【０２１３】［４］リトライ処理

前記［３．３］において、ブリッジキャッシュのエビクションが必要な場合におけるトランザクションのリトライについて説明したが、ここではその他にリトライを行う場合についての構成について説明する。

【０２１４】［４．１］スプリット中に上位バスに別のトランザクションが発行された場合

バスブリッジが上位バス・トランザクションをスプリットし、スプリットにより下位バスに発行したトランザクションに対する下位バスからの応答がバスブリッジに来る前に、別のトランザクションが上位バスに発行された場合、先に発行されたトランザクションが使用する予定のキャッシュブロックが後に発行されたトランザクションによって使用されてしまい、前に発行されたトランザクションに対する下位バスからの応答をブリッジキャッシュに格納できなくなる可能性がある。

【０２１５】このような問題を解消するためにまず考えられるのは、バスブリッジにスプリット中か否かを示すフラグを設け、上位バス・トランザクションを受け取ったときにフラグがスプリット中を示していれば、そのトランザクションのスプリットを認めないようにすることである。すなわち、バスブリッジは、トランザクションを受け取ると、フラグを調べるとともにブリッジキャッシュをスヌープし、スプリット中かつ受け取ったトランザクションのスプリットが必要と分かった場合に、トランザクションをリトライさせる。このようにすれば、スプリット中のトランザクションのために確保されたキャッシュブロックが、後から発行されたトランザクションによって使用されるのを防ぐことができる。

【０２１６】しかしながら、この方法では、他のトランザクションのスプリット中に発行されたトランザクションは、必ずリトライするので無駄が多い。そこで、リトライを効率的に行う方法として、本実施形態では次の２つの方法を採用する。

【０２１７】第１の方法は、スプリット中にトランザクションが発行された場合、そのトランザクションの種類から、そのトランザクションをスプリットしたときにブリッジキャッシュのブロックを使用するかどうかを判定し、ブロックを使用する場合にのみリトライを行うという方法である。例えば、Ｒｅａｄトランザクションは、スプリットした場合に、下位バスからの応答データを受け入れるためブロックを使用するが、Ｉｎｖａｌｉｄａｔｅトランザクションは、スプリットしたとしてもブロックを使用しない。このようなトランザクションの種類によってリトライを行うか否かを判別することにより、無駄なリトライを減らすことができる。

【０２１８】この構成では、バスブリッジは、まず上位バス・トランザクションを受け取ると、フラグを調べて現在他のトランザクションがスプリット中か否かを判定

する。そして、スプリット中の場合は、上位バスから受け取ったトランザクションの種類を検出して、その種類がブロックを使用するものか否かを判定する。この結果、ブロックを使用するトランザクションである場合には、そのトランザクションのスプリットを認めない。すなわち、この場合、ブリッジキャッシュのスヌープによりそのトランザクションのスプリットが必要と判定されると、バスブリッジから当該トランザクションについてリトライ終了信号を返す。

【０２１９】この構成によれば、キャッシュブロックを使用することのないトランザクションは、リトライせずにそのまま実行することができる。

【０２２０】また、第２の方法は、スプリット中のトランザクションの使用予定ブロックのアドレスをバスブリッジで管理し、使用予定ブロックに対するトランザクションのみリトライするという方法である。すなわち、この方法では、スプリット中に新たに発行されたトランザクションが、現在スプリット中のトランザクションが使用する予定のブロックを使用するか否かを判定し、使用すると判定される場合には新たに受け取ったトランザクションに対してリトライを指示する。

【０２２１】図５は、このようなリトライ判定機能を有するバスブリッジの概略構成を示した図である。図５において、図３と同一の部分には同一の符号を付してその説明を省略する。

【０２２２】図５においては、バスブリッジ３０内に現在スプリット中のトランザクションが使用するキャッシュブロックのアドレス情報を格納するスプリット情報レジスタ７２が設けられている。

【０２２３】まず、図５の構成におけるスプリット情報レジスタ７２へのアドレス情報の登録動作について説明する。この構成において、バスブリッジ制御部６０は、上位バス２０から受け取ったトランザクションをスプリットすると決定した場合、当該トランザクションのアドレスに対応するキャッシュブロックのアドレス情報を求め、スプリット情報レジスタ７２に登録する。スプリット情報レジスタ７２に登録されるアドレス情報は、例えばブリッジキャッシュ３１がセットアソシアティブ方式である場合は、該当ブロックのインデックス及びウエイ番号である。そして、スプリットにより下位バス４０に発行したトランザクションに対する応答が戻り、バスブリッジ制御部６０から上位バス２０に対してスプリット応答トランザクションを発行した時点で、当該トランザクションに対するスプリット情報レジスタ７２の登録情報は消去される。

【０２２４】したがって、スプリット情報レジスタ７２には、現在スプリット中のトランザクションが使用する予定のブロックのアドレス情報が格納されていることになる。なお、スプリット情報レジスタ７２は複数のアドレス情報が登録可能であり、同時に複数のトランザクションのスプリットに対応可能となっている。

【０２２５】次に、スプリット情報レジスタ７２を用いた上位バス・トランザクションのリトライ判定について説明する。

【０２２６】上位バス・トランザクションを受け取ったバスブリッジ制御部６０は、そのトランザクションのアドレスから、対応するキャッシュブロックのアドレス情報（例えば、インデックス及びウエイ番号）を求め、このアドレス情報をスプリット情報レジスタ７２に登録されているアドレス情報と比較する。そして、この比較の結果、両者が一致した場合には、バスブリッジ制御部６０は、上位バス・トランザクションに対する応答としてリトライ終了信号を返す。一方、比較の結果両者が一致しない場合には、バスブリッジ制御部６０は、通常のトランザクション処理を行う。

【０２２７】このように、図５に示すバスブリッジ３０によれば、スプリット中のトランザクションが使用する予定のキャッシュブロックに対するトランザクションは、リトライせずにそのまま実行することができる。

【０２２８】なお、スプリット情報レジスタ７２にスプリットしたトランザクションのアドレス情報をすべて登録するかわりに、スプリットしたトランザクションのうちブリッジキャッシュを使用するトランザクションを選別し、このようなトランザクションのアドレス情報のみを登録する構成とすれば、無駄なリトライをさらに減らすことができる。なお、ブリッジキャッシュを使用するトランザクションは、トランザクションの種類から判別する。

【０２２９】また、前記第１及び第２の方法を併用すればリトライのさらなる効率化を達成することができる。

【０２３０】［４．２］エビクションバッファに退避中のデータに対するトランザクションが発行された場合

この状況は、前記［３．３］（ｂ）（図３参照）で説明したエビクションバッファを用いた構成において生じる。すなわち、エビクション対象のブロックをエビクションバッファに退避し、まだエビクションの処理が完了しないうちに、退避したデータに対するトランザクションが上位バスに発行された場合、エビクションバッファに格納されたデータにはアクセス不可能なので、そのような退避データに対するトランザクションはリトライする必要がある。この場合、エビクションバッファにデータを退避させている間に上位バスに発行されたトランザクションをすべてリトライさせるという方法もあるが、これではリトライが多くなり処理効率が向上しない。

【０２３１】そこで、本実施形態では、このような場合のリトライの発行を効率化するために、バスブリッジに、エビクションバッファ内のデータのアドレスとトランザクションのアドレスとを比較する手段を設ける。そして、比較の結果、トランザクションのアドレスと一致するアドレスのデータがエビクションバッファに格納さ

れているとわかった場合には、そのトランザクションをリトライさせる。

【０２３２】図３を参照して、この構成におけるバスブリッジの動作を説明する。上位バス２０にトランザクションが発行された場合、バスブリッジ制御部６０は、そのトランザクションのアドレスと、エビクションバッファ６２のアドレスレジスタ６４に登録されているアドレスとを比較する（このアドレスの比較は、ブロック単位の比較、すなわちタグ、インデックスまでの比較である）。この結果、トランザクションのアドレスと同一のアドレスがアドレスレジスタに登録されていた場合には、バスブリッジ制御部６０は上位バス２０に対してリトライ終了信号を出力する。

【０２３３】この構成によれば、エビクションバッファを採用した構成において、そのエビクションバッファに退避中のデータに対するトランザクションのリトライ処理を効率的に行うことができる。

【０２３４】［４．３］下位バスからのリトライ指示に対するバスブリッジの対応

リトライの指示は、下位バスのＩ／Ｏ装置などからも発せられる。このような下位バスからのリトライ指示をバスブリッジが受け取った場合、バスブリッジが所定時間後にトランザクションの再発行を行うという構成も考えられるが、この場合、バスブリッジの構成が複雑となってしまう。

【０２３５】そこで、バスブリッジの構成を簡単化するためには、バスブリッジは下位バスからリトライ終了信号を受け取った場合に、そのリトライ終了信号を上位バスに転送する構成を採用することもできる。すなわち、バスブリッジは、下位バスからリトライ終了信号を受け取った場合、スプリット応答トランザクションを生成して上位バスに発行し、そのスプリット応答トランザクションにてリトライ終了信号を出力する。そして、そのリトライ終了信号を受け取ったトランザクション発行元ＣＰＵが、当該トランザクションのリトライを行う。

【０２３６】この構成によれば、下位バスからのリトライ指示に対するバスブリッジの処理が簡単化される。

【０２３７】以上、本実施形態の階層バスシステムの動作について説明した。以下では、本実施形態におけるバスブリッジの構成及びその動作について説明する。

【０２３８】図６は、本実施形態におけるバスブリッジ及びブリッジキャッシュの構成を示すブロック図である。

【０２３９】図６において、バスブリッジ３００は、信号線５００～５１０によって上位バス２０と信号のやり取りを行い、信号線５１２～５２２によって下位バス４０と信号のやり取りを行う。

【０２４０】また、バスブリッジ３００のブリッジキャッシュは、図６においては、タグＲＡＭ４０２及びキャッシュＲＡＭ４０４から構成されている。キャッシュＲＡＭ４０４は、データをブロック単位で格納する。メモリから読み込まれたデータは、そのメモリアドレスのインデックス部に対応するアドレスのブロックに格納される。一方、タグＲＡＭ４０２は、キャッシュＲＡＭ４０４の各ブロックごとにメモリアドレスのタグ部を格納しており、キャッシュＲＡＭ４０４に各ブロックに格納されているデータがどのメモリアドレスのデータであるかを示している。また、タグＲＡＭ４０２は、各ブロックごとに、そのブロックに格納されているデータのＭＥＳＩステートを保持している。

【０２４１】次に、図７に示す構成の階層バスシステムを例にとって、図６に示したバスブリッジ及びブリッジキャッシュの動作例を説明する。ここで、図７のバスブリッジ＃０及び＃１と、ブリッジキャッシュ＃０及び＃１とは、それぞれ図６に示す構成を有しているものとする。また、この例においては、ブリッジキャッシュはダイレクトマッピング方式のキャッシュであるとして説明する。

【０２４２】この動作例の初期状態を以下に示す。

【０２４３】

ＣＰＵ＃０：　アドレスＡのデータを持っていない（アドレスＡに対応するＣＰＵキャッシュのキャッシュブロックは無効化されている）

ブリッジキャッシュ＃０：　アドレスＡのデータを持っていない。アドレスＡに対応するブロックにはアドレスＢのデータがステートＭで格納されている

ＣＰＵ＃１：　アドレスＢのデータをステートＭで持っている

ブリッジキャッシュ＃１：　アドレスＡのデータをステートＭで持っている

ＣＰＵ＃３：　アドレスＡのデータをステートＭで持っている

以下、このような初期状態から、ＣＰＵ＃０がアドレスＡをリードするリクエストを発した場合のシステムの動作について説明する。

【０２４４】（１）ＣＰＵ＃０は、自らのキャッシュを検索するもののアドレスＡのデータが存在しないので、上位バス＃０にＲｅａｄトランザクション（アドレスＡ）を発行する。

【０２４５】（２）バスブリッジ＃０は、上位バス＃０から信号線５０４を経由してトランザクションの種類（Ｒｅａｄ）及びアドレス（Ａ）を得る。これらの情報は、上位トランザクションバッファ３１２、タグリード回路３１８、下位バス出力判定回路３１４に入力される。すなわち、(2.1) 上位トランザクションバッファ３１２は、入力された上位バス・トランザクションの種類、アドレス及びスプリットＩＤなど上位バストランザクションに関する諸情報を格納する。なお、スプリットＩＤとは、バス上のトランザクションを識別するためのＩＤであり、トランザクションを発したエージェント

（ＣＰＵ、バスブリッジ等）のＩＤと当該エージェント内でのトランザクションのＩＤを含んでいる。

【０２４６】すなわち、この動作例では、トランザクション種類“Ｒｅａｄ”、アドレス“Ａ”及びＲｅａｄに割り当てられたスプリットＩＤなどの情報が上位トランザクションバッファ３１２に格納される。

【０２４７】(2.2) タグリード回路３１８は、入力されたアドレスによって指定されるデータがブリッジキャッシュ内に存在するか否かを調べる。このとき、まず、タグリード回路３１８は、入力されたアドレスからインデックスを抽出し、セレクタ３５０を介してタグＲＡＭ４０２に入力する。タグＲＡＭ４０２は、入力されたインデックスに対応するブロックのタグ情報及びステートをタグリード回路３１８に返す。

【０２４８】タグリード回路３１８は、タグＲＡＭ４０２から返されてきたタグとトランザクション・アドレスのタグ部とを比較し、両者の一致・不一致を判定する。そして、両者が一致する場合には該当ブロックのステートを下位バス出力判定回路３１４に出力する。この場合、当該ブロックのステートがＭ又はＥ又はＳであれば、キャッシュＲＡＭ４０４内にＲｅａｄの要求するデータがあるということを意味する。

【０２４９】一方、両者が不一致の場合は、キャッシュＲＡＭ４０４内にはＲｅａｄが要求するデータはないということなので、タグリード回路３１８はＲｅａｄの要求するアドレスＡに対するブリッジキャッシュのステートとして、ステートＩを下位バス出力判定回路３１４に出力する。ただし、タグが不一致の場合には、該当ブロックに格納されている別アドレスのデータがどのステートであるかによってエビクションが必要となる場合があるので、タグリード回路３１８は、タグＲＡＭ４０２から受け取った該当ブロック自体のステートも下位バス出力判定回路３１４に出力する。

【０２５０】この動作例では、ブリッジキャッシュ＃０にはアドレスＡに対応するブロックにアドレスＢのデータがステートＭで格納されているので、タグリード回路３１８から下位バス出力判定回路３１４へは、Ｒｅａｄの要求データ（アドレスＡ）が存在しないことを示すステートＩの信号と、該当ブロック自体のステートＭの信号が出力される。

【０２５１】(2.3) 下位バス出力判定回路３１４は、信号線５０４から入力されたトランザクションの種類と、タグリード回路３１８から入力されたトランザクション・アドレスのステートとに基づき、下位バスに出力すべきトランザクションを決定する。このとき、下位バス出力判定回路３１４は、制約１を採用する場合には表３又は４の規則に従い、制約２を採用する場合は表５の規則に従って、下位バスに出力するトランザクションを決定する。

【０２５２】この動作例では、上位バス・トランザクションはＲｅａｄであり、Ｒｅａｄのアドレスに対するブリッジキャッシュ＃０のステートはＩなので、制約１及び２のいずれを採用した場合でも、下位バス出力判定回路３１４は上位バスのトランザクションをスプリットし、下位バスに対してアドレスＡのデータを要求するＲｅａｄを発行すると決定する。

【０２５３】下位バス出力判定回路３１４の判定結果は、下位バストランザクション生成・保持回路３３０に入力される。なお、この場合は、上位バスのＲｅａｄはスプリットされるので、下位バス出力判定回路３１４は、スプリットを行ったことを示す情報を出力する。この情報は上位バストランザクションバッファ３１２に前記トランザクションの種類及びアドレス等と対応づけて登録される。また、この情報は、レスポンス生成回路３１０にも出力される。

【０２５４】(2.4) 下位バストランザクション生成・保持回路３３０は、与えられた判定結果に従って、アドレスＡのデータを要求するＲｅａｄトランザクションを生成する。同時に、下位バストランザクション生成・保持回路３３０は、下位バスアービトレーション回路３２８に所定の信号を発し、これを受けた下位バスアービトレーション回路３２８は、下位バスの使用権を得るために信号線５１４から下位バスに対してアービトレーション信号を発する。この例では各バスは分散アービトレーション方式を採用しており、下位バスアービトレーション回路３２８は、下位バスアービトレーション回路３２８が発したアービトレーション信号に対する他のバスブリッジ＃１のバス要求状態を信号線５１２から得、自バスブリッジ＃０が下位バスに対してトランザクションを発行できるか否かを判定する。そして、下位バスアービトレーション回路３２８は、下位バスに対してトランザクションが発行可能となったときに出力バッファ３６２に対して所定の信号を発し、この信号により、下位バストランザクション生成・保持回路３３０で生成されたＲｅａｄが出力バッファ３６２から信号線５１６を経由して下位バスに出力される。

【０２５５】(2.5) このようにしてバスブリッジ＃０が下位バスに対して発行したＲｅａｄは、下位バスから信号線５１６を経由してバスブリッジ＃０自身に取り込まれ、そのトランザクション種類（Ｒｅａｄ）及びアドレス（Ａ）が下位バストランザクションバッファ３２６に格納される。

【０２５６】(2.6) レスポンス生成回路３１０は、下位バス出力判定回路３１４からスプリットを行ったことを示す信号を受取り、これに基づきスプリット終了信号を生成し、上位バス＃０に出力する。なお、ＣＰＵ＃０は、このスプリット終了信号を受取ると、いったん上位バス＃０を解放し、下位バスに発行したＲｅａｄの結果がバスブリッジ＃０から返ってくるのを待つ。

【０２５７】（３）バスブリッジ＃０は、前記（２）の

処理と同時に、ブリッジキャッシュ＃０のエビクションが必要か否かを判定する。エビクションが必要と判定された場合、ブリッジキャッシュ＃０の該当ブロックのデータをキャッシュＲＡＭ４０４からエビクションバッファ３３６に退避させる。また、バスブリッジ＃０は、エビクションが必要な場合は、上位バス＃０へのＩｎｖａｌｉｄａｔｅトランザクションを生成し、さらに下位バスへのＷｒｉｔｅＢａｃｋトランザクションの発行の要否を判定する。すなわち、(3.1) 下位バス出力判定回路３１４は、前記(2.3) におけるタグリード回路３１８からの出力に基づきエビクションの要否を判定する。エビクションが必要となるのは、トランザクションのアドレスに対応するキャッシュブロックが、他のアドレスによって使用されている（当該他のアドレスについてのステートがM又はＥ又はＳである）場合である。したがって、下位バス出力判定回路３１４は、トランザクションに対するステートがＩであり、該当ブロック自体のステートがM又はＥ又はＳであった場合には、ブリッジキャッシュの該当ブロックのエビクションが必要と判定する。この判定結果は、キャッシュ制御回路３２２に入力される。なお、下位バス出力判定回路３１４からキャッシュ制御回路３２２に与えられる出力には、エビクションすべきブロックのアドレス情報が含まれる。

【０２５８】この動作例では、初期状態として、ブリッジキャッシュ＃０において、アドレスＡに対応するブロックにはアドレスＢのデータがステートMで格納されている。したがって、下位バス出力判定回路３１４は、ブリッジキャッシュ＃０のエビクションが必要と判定し、アドレスＢをキャッシュ制御回路３２２に出力する。

【０２５９】(3.2) 下位バス出力判定回路３１４の出力を受け取ったキャッシュ制御回路３２２は、その出力に含まれるアドレス情報をキャッシュＲＡＭ４０４に入力する。キャッシュＲＡＭ４０４は、そのアドレス情報に基づき、エビクション対象ブロックのデータをエビクションバッファ３３６に出力し、当該ブロックを空き状態にする。エビクションバッファ３３６は、キャッシュＲＡＭ４０４から受け取ったデータを格納する。

【０２６０】また、キャッシュ制御回路３２２は、前記アドレス情報をエビクションバッファ３３６に入力する。エビクションバッファ３３６は、このアドレス情報を、前記キャッシュＲＡＭ４０４から受け取ったデータに対応付けて記憶する。

【０２６１】この動作例では、アドレスＢに対応するブロックのデータがキャッシュＲＡＭ４０４からエビクションバッファ３３６に掃き出され、キャッシュＲＡＭ４０４はそのデータをアドレスＢと対応付けて格納する。

【０２６２】この結果、キャッシュＲＡＭ４０４のアドレスＢに対応するブロックは解放され、Ｒｅａｄトランザクションが使用できる状態となる。

【０２６３】このように、図６の構成では、エビクション対象のブロックをエビクションバッファ３３６に退避することにより、上位バスから受け取ったトランザクションのために当該ブロックを使用することができる。したがって、この構成では、エビクションに伴うＩｎｖａｌｉｄａｔｅ及びＷｒｉｔｅＢａｃｋの処理の完了を待たずに、下位バスへのトランザクションの発行等のトランザクションに対する通常処理を進行することが可能となる。

【０２６４】(3.3) 下位バス出力判定回路３１４は、エビクションが必要と判定した場合には、タグリード回路３１８から与えられるエビクション対象ブロックのステート情報に基づき、下位バスにＷｒｉｔｅＢａｃｋを発行するか否かを判定する。この判定は、制約１を採用した場合には表１６の規則に従い、制約２を採用した場合には表１７の規則に従って行われる。

【０２６５】この動作例では、Ｒｅａｄのアドレスに対応するブロックにはアドレスＢのデータがステートMで格納されているので、いずれの制約条件を採用する場合でも、下位バスにＷｒｉｔｅＢａｃｋを行うことが決定される。

【０２６６】この判定結果は、上位バストランザクションバッファ３１２に入力され、対応トランザクションの情報に対応づけて登録される。

【０２６７】なお、この判定に基づくＷｒｉｔｅＢａｃｋの発行は、次に説明する上位バス＃０へのＩｎｖａｌｉｄａｔｅが完了した後に行われる。

【０２６８】(3.4) 上位バス出力判定回路３１６は、前記(2.3) におけるタグリード回路３１８からの出力に基づき、前記下位バス出力判定回路３１４と同様にして、エビクションの要否を判定する。ここでエビクションが必要と判定された場合は、上位バス出力判定回路３１６は、エビクション対象のブロックのステート（すなわち、該当ブロックに格納されているデータのステート）に基づき、前記表１５の規則に従って上位バスへのＩｎｖａｌｉｄａｔｅの発行を決定する。そして、上位バス出力判定回路３１６は、この決定に応じて、Ｉｎｖａｌｉｄａｔｅの発行を指示する信号を上位バストランザクション生成・保持回路３０４に出力する。

【０２６９】この信号を受け取った上位バストランザクション生成・保持回路３０４は、Ｉｎｖａｌｉｄａｔｅトランザクションを生成すると共に、上位バスアービトレーション回路３０２に対してアービトレーションを指示する。そして、上位バス＃０の使用権が得られ次第、上位バストランザクション生成・保持回路３０４から出力バッファ３６０を介して、上位バス＃０にＩｎｖａｌｉｄａｔｅが発行される。

【０２７０】この動作例では、上位バス出力判定回路３１６は、エビクションが必要と判定する。この結果、上位バストランザクション生成保持回路３０４にてＩｎｖａｌｉｄａｔｅが生成され、上位バス＃０のエビクショ

ンが完了し次第発行される。したがって、例えばパイプライン処理が可能なシステムとすれば、ＣＰＵ＃０からのＲｅａｄの発行のサイクルが終わり次第、バスブリッジ＃０から上位バス＃０に対してＩｎｖａｌｉｄａｔｅの発行が可能となる。

【０２７１】以上に説明した（３）の処理は、前記（２）の処理と同時並行して行われる。そして、これ以降、本システムでは、（２）に続く通常のトランザクション処理（（４）～（８））と、（３）に続くエビクション処理（（９）～（１１））とを行う。本システムにおいてパイプライン方式のアーキテクチャを採用すれば、これら通常処理とエビクション処理とは同時並行的に行うことも可能である。

【０２７２】（４）バスブリッジ＃１は、前記（２）で下位バスに発行されたトランザクションをモニタする。すなわち、バスブリッジ＃１は、信号線５１６から下位バス・トランザクションの種類（Ｒｅａｄ）及びアドレス（Ａ）を得て、これらの情報を下位トランザクションバッファ３２６、タグリード回路３１８、上位バス出力判定回路３１６に入力する。すなわち、(4.1) 下位トランザクションバッファ３２６は、入力された下位バス・トランザクションの種類、アドレス及び当該トランザクションを発行したバスブリッジのＩＤなど下位バストランザクションに関する諸情報を格納する。

【０２７３】この動作例では、トランザクション種類“Ｒｅａｄ”、アドレス“Ａ”等の情報が下位バストランザクションバッファ３２６に格納される。

【０２７４】(4.2) タグリード回路３１８は、前述のステップ(2.1) と同様にしてタグＲＡＭ４０２を検索し、下位バス・トランザクションの要求データに対するブリッジキャッシュ＃１のステートを調べ、そのステートを上位バス出力判定回路３１６に出力する。

【０２７５】この例では、ブリッジキャッシュ＃１は、Ｒｅａｄの要求データをステートＭで所有しているので、タグリード回路３１８からは、ステートＭを示す信号が上位バス出力判定回路３１６に与えられる。

【０２７６】(4.3) 上位バス出力判定回路３１６は、信号線５１６から入力されたトランザクション種類とタグリード回路３１８から入力されたトランザクションに対するステートの情報とから、上位バス＃１に発行するトランザクションを決定する。ここで、上位バス＃１に発行するトランザクションの決定においては、制約１を採用する場合には表９の規則に従い、制約２を採用する場合には表１０の規則に従う。この動作例では、いずれの制約条件を採用した場合でも、Ｒｅａｄの発行が決定される。この上位バス出力判定回路３１６の判定結果は、上位バストランザクション生成・保持回路３０４に入力される。

【０２７７】(4.4) 上位バストランザクション生成・保持回路３０４は、与えられた判定結果に従って、アドレスＡのデータを要求するＲｅａｄトランザクションを生成する。同時に、上位バストランザクション生成・保持回路３０４は、上位バスアービトレーション回路３０２に所定の信号を発し、これを受けた上位バスアービトレーション回路３０２は、上位バス＃１の使用権を得るために信号線５００から上位バス＃１に対してアービトレーション信号を発する。そして、上位バスアービトレーション回路３２８は、このアービトレーション信号に対する他のバスブリッジのバス要求状態を信号線５０２から得、バスブリッジ＃１が上位バス＃１に対してトランザクションを発行できるか否かを判定する。そして、上位バス＃１に対してトランザクションが発行可能となったときに、上位バスアービトレーション回路３０２は出力バッファ３６０に対して所定の信号を発し、この結果上位バストランザクション生成・保持回路３０４で生成されたＲｅａｄが出力バッファ３６０から信号線５０４を経由して上位バス＃１に発行される。

【０２７８】(4.5) このようにしてバスブリッジ＃１が上位バスに対して発行したＲｅａｄは、上位バスから信号線５０４を経由してバスブリッジ＃１自身に取り込まれ、そのトランザクション種類（Ｒｅａｄ）及びアドレス（Ａ）が上位バストランザクションバッファ３１２に格納される。

【０２７９】（５）上位バス＃１に接続されたＣＰＵ＃２及び＃３は、バスブリッジ＃１から上位バス＃１に発行されたＲｅａｄをモニタする。ここで、ＣＰＵ＃３はキャッシュ内にアドレスＡのデータをステートＭで持っているので、ＣＰＵ＃３は当該Ｒｅａｄに対してＨＩＴＭをスヌープ出力するとともに、アドレスＡに対応するキャッシュブロックのデータを上位バス＃１に出力する。

【０２８０】（６）バスブリッジ＃１は、ＣＰＵ＃３から出力されたステートＭのデータによって自身のブリッジキャッシュの該当ブロックを更新すると共に下位バスにそのデータを出力し、さらに下位バスに対してＨＩＴＭをスヌープ出力する。そして、バスブリッジ＃１は、自身のブリッジキャッシュの該当ブロックのステートを変更する。すなわち、(6.1) ＣＰＵ＃３から出力されたステートＭのデータは、信号線５１０を経由してバスブリッジ＃１に入力される。このデータは、セレクタ３５２を介してキャッシュＲＡＭ４０４に入力されると共に、信号線５２２を経由して下位バスに出力される。ここで、キャッシュＲＡＭ４０４の該当ブロックのデータ内容は前記ＣＰＵ＃３からのデータによって更新される。このときのキャッシュＲＡＭ４０４内における更新ブロックの特定は次のようにして行われる。すなわち、ステップ(4.5) に示したように、バスブリッジ＃１は、自らが上位バスに対して発行したＲｅａｄを信号線５０４を経由して取り込むが、この上位バスＲｅａｄトランザクションのアドレスが下位バス出力判定回路３１４を

介してキャッシュ制御回路３２２に入力される。キャッシュ制御回路３２２は、このアドレス情報からインデックスを抽出し、このインデックス情報をキャッシュＲＡＭ４０４に与えることにより更新すべきブロックを特定する。このようにして特定されたブロックに対して信号線５１０から入力されたデータが上書きされる。なお、信号線５２２から下位バスに出力されたデータは、バスブリッジ＃０によって取り込まれると共に、メインメモリにも入力され、データ内容の更新が行われる。

【０２８１】(6.2) バスブリッジ＃１から下位バスに対するスヌープ出力は、下位バススヌープ出力生成回路３３２にて生成される。下位バススヌープ出力生成回路３３２には、信号線５０６からの上位バスのスヌープ結果（上位の各ＣＰＵのスヌープ出力のワイヤードＯＲ）と、ステップ(4.1) で下位バストランザクションバッファ３３０に格納された下位バス・トランザクションの種類と、上位バス出力判定回路３１６がステップ(4.2) で検出したブリッジキャッシュのステートと、が入力される。ここで、制約１を採用した場合、下位バススヌープ出力生成回路３３２は、下位バストランザクションの種類とブリッジキャッシュのステートとの組み合わせから表１１の規則に従ってスヌープ出力を生成する。また、制約２を採用した場合は、下位バススヌープ出力生成回路３３２は、下位バストランザクションの種類、ブリッジキャッシュのステート及び上位バススヌープ結果の組み合わせから表１２の規則に従ってスヌープ出力を生成する。このようにして生成されたスヌープ出力は、信号線５１８を経由して下位バスに出力される。

【０２８２】この動作例では、下位バストランザクションの種類がＲｅａｄで、ブリッジキャッシュのステートがMなので、いずれの制約条件を採用する場合でも、下位バススヌープ出力生成回路３３２はＨＩＴＭを出力する。

【０２８３】(6.3) ブリッジキャッシュの次ステートはタグ更新回路３２４で決定される。タグ更新回路３２４には、信号線５０６からの上位バススヌープ結果と、ステップ(4.5) において上位バストランザクションバッファ３１２に格納された上位バス・トランザクションの種類が与えられる。タグ更新回路３２４は、これらの情報に基づき表１３又は表１４の規則に基づいて、ブリッジキャッシュの次ステートを決定する。この例では、上位バススヌープ結果がＨＩＴＭで上位バストランザクションがＲｅａｄなので、ブリッジキャッシュの次ステートはＳと決定される。

【０２８４】また、タグ更新回路３２４は、上位バストランザクションバッファ３１２からトランザクションのアドレスＡを得、このアドレスＡと前記次ステートＳとをタグライト回路３２０に入力する。タグライト回路３２０は、このアドレスＡのインデックス部をタグＲＡＭ４０４に入力してタグ情報の書き換えを行うべきブロックを指定し、さらに前記次ステートＳをタグＲＡＭ４０２に入力して当該ブロックのステートを変更する。

【０２８５】（７）バスブリッジ＃０は、下位バスに発行したトランザクションに対する他のバスブリッジからの応答を検出し、上位バス＃０に対する応答の生成を行う。この例では、下位バス上にはバスブリッジ＃０及びバスブリッジ＃１が接続されているだけなので、バスブリッジ＃０は、前記（６）でバスブリッジ＃１から下位バスに出力されたデータやスヌープ出力などを検出すると、それらに基づき次のような処理を行う。

【０２８６】まず、バスブリッジ＃０は、ＣＰＵ＃０に応答を行うためにスプリット応答トランザクションを生成し、上位バス＃０上に発行する。また、バスブリッジ＃０は、ＣＰＵ＃０の次ステートを制御するために、スヌープ出力を生成して上位バス＃０に出力する。また、バスブリッジ＃０は、下位バスから得られたデータをブリッジキャッシュ＃０に書き込む。さらに、バスブリッジ＃０は、自らのブリッジキャッシュのステートを変更する。すなわち、(7.1) スプリット応答生成回路３０６は、信号線５１８から下位バススヌープ結果が入力されると、上位バストランザクション生成・保持回路３０４に対してスプリット応答トランザクション生成指令を発する。すなわち、スプリット応答生成回路３０６は、下位バススヌープ結果を、スプリット応答トランザクションを生成するためのタイミング信号として用いている。スプリット応答生成回路３０６は、下位バススヌープ結果を得たときに、上位トランザクションバッファ３１２に格納されているトランザクション情報に基づき、トランザクション発行元のＣＰＵに対してスプリット応答トランザクション生成指令を生成する。上位トランザクションバッファ３１２には、上位バスから受け取ったトランザクションの情報（種類、アドレス、スプリットＩＤ等）がスプリットを行ったか否かの情報と共に格納されている。スプリット応答生成回路３０６は、この上位トランザクションバッファ３１２からスプリットを行ったトランザクションを検索し、そのトランザクションのスプリットＩＤからそのトランザクションを発行したＣＰＵのＩＤを検出して、当該ＣＰＵ＿ＩＤに対するスプリット応答トランザクションを生成する。

【０２８７】この動作例では、バスブリッジ＃１から出力されたＨＩＴＭの信号が信号線５１８から入力される。このＨＩＴＭの信号を受けると、スプリット応答生成回路３０６は、上位トランザクションバッファ３１２からスプリットを行ったトランザクションを検出し、そのトランザクションの発行元ＣＰＵのＩＤ（スプリットＩＤに含まれている。ここでは＃０）を検出し、このＣＰＵ＃０に対するスプリット応答トランザクションの生成指令を上位バストランザクション生成・保持回路３０４に出力する。

【０２８８】上位バストランザクション生成・保持回路

３０４は、この指令に基づいてスプリット応答トランザクションを生成する。そして、上位バスアービトレーション回路３０２にて上位バスの使用権を獲得したところで、信号線５０４から上位バス＃０にそのスプリット応答トランザクションが発行される。

【０２８９】(7.2) バスブリッジ＃０は、信号線５１８から得た下位バススヌープ結果と、前記ステップ(2.5)で下位バストランザクションバッファ３２６に格納された下位バス・トランザクション種類とに基づき、上位バスに対するスヌープ出力を生成する。このスヌープ出力の生成は、制約１を採用する場合には前記表７の規則に従って行い、制約２を採用する場合には前記表８の規則に従って行う。この処理は上位バススヌープ出力生成回路３０８で行われ、生成されたスヌープ出力は信号線５０６を介して上位バス＃０に出力される。この例では、スヌープ出力としてＨＩＴが上位バス＃０に出力される。

【０２９０】(7.3) バスブリッジ＃０は、バスブリッジ＃１から下位バスに出力された最新のキャッシュデータを信号線５２２を経由して受信し、これをキャッシュＲＡＭ４０４に格納する。すなわち、ステップ(7.1)にて上位バス上に発行されたスプリット応答トランザクションはバスブリッジ＃０自身にて受信され、このとき下位バス出力判定回路３１４が上位トランザクションバッファ３１２に格納されているアドレスを読出してキャッシュ制御回路３２２に与える。そして、キャッシュ制御回路３２２は、このアドレスからインデックスを抽出し、このインデックス情報をキャッシュＲＡＭ４０４に与えることにより書き込み対象のブロックを特定する。そして、キャッシュＲＡＭ４０４では、キャッシュ制御回路３２２によって指定されたブロックに対して、信号線５２２から入力されたデータを書き込む。なお、このようにしてキャッシュＲＡＭ４０４に格納された最新データは、続いてキャッシュＲＡＭ４０４から信号線５１０を介して上位バス＃０に出力される。

【０２９１】この例では、バスブリッジ＃０は、下位バスからの応答によって得たアドレスＡの最新データをブリッジキャッシュ＃０に書き込むと共に、上位バス＃１に出力する。

【０２９２】(7.4) タグ更新回路３２４は、信号線５１８から得られる下位バススヌープ結果と、ステップ(2.5)で下位バストランザクションバッファ３２６に格納された下位バストランザクションの種類とに基づき、ブリッジキャッシュの次ステートを決定する。この次ステートの決定は、制約１及び制約２のいずれを採用する場合でも。前記表６の規則に従って行う。この例では、下位バススヌープ結果がＨＩＴＭで下位バストランザクションがＲｅａｄなので、次ステートはＳと決定される。

【０２９３】また、タグ更新回路３２４は、下位バストランザクションバッファ３１２から下位バスに発行したＲｅａｄトランザクションのアドレスを得、このアドレスと前記次ステートとをタグライト回路３２０に入力する。タグライト回路３２０は、このアドレスのインデックス部をタグＲＡＭ４０２に入力して書き換えを行うべきブロックを指定し、さらにアドレスのタグ部及び前記次ステートをタグＲＡＭ４０２に入力して当該ブロックのタグ及びステートを更新する。

【０２９４】（８）前記（２）におけるスプリットの結果、いったん上位バス＃０を放棄したＣＰＵ＃０は、バスブリッジ＃０から上位バス＃０に発行されたスプリット応答トランザクションを受取り、そのスプリット応答トランザクションに付随するデータを、ＣＰＵキャッシュの該当ブロックに書き込む。また、このとき、バスブリッジ＃０からのスヌープ出力ＨＩＴによりＣＰＵ＃０の受け取るスヌープ結果はＨＩＴとなり、この結果、ＣＰＵ＃０は当該ブロックのステートをＭＥＳＩプロトコルにしたがってＳに変更する。

【０２９５】（９）ここからは、前記（３）の処理に続くエビクションの処理について説明する。この（９）～（１１）の処理は、必ずしも前記（４）～（８）の処理の後に来るものではなく、両者の処理は並行的に行うこともできる。

【０２９６】前記ステップ(3.4)にて発行されたＩｎｖａｌｉｄａｔｅを受け取った上位バス＃０上の各ＣＰＵは、当該Ｉｎｖａｌｉｄａｔｅの対象ブロックを無効化する。この例では、Ｉｎｖａｌｉｄａｔｅはアドレスに対するものであり、ＣＰＵ＃１はアドレスＢのデータをステートＭで持っている。したがって、ＣＰＵ＃１は、Ｉｎｖａｌｉｄａｔｅに対する応答として、上位バス＃０に対してＨＩＴＭを出力し、アドレスＢのブロックのステートＭのデータを上位バス＃０に出力する。そして、ＣＰＵ＃０は、キャッシュの当該ブロックのステートをＭからＩに変更する。

【０２９７】（１０）バスブリッジ＃０は、Ｉｎｖａｌｉｄａｔｅに対応してＣＰＵ＃１から書き戻されたデータを信号線５１０を経由して受取り、このデータによりエビクションバッファ３３６の内容を更新する。すなわち、エビクションバッファ３３６には、すでにステップ(3.2)でキャッシュＲＡＭ４０４に格納されていたアドレスＢのデータが退避しているが、このデータに対して、ＣＰＵ＃１から書き戻されてきたアドレスＢの最新データが上書きされる。

【０２９８】（１１）(11.1)バスブリッジ＃０は、前記ステップ(3.3)にて自らが上位バス＃０に発行したＩｎｖａｌｉｄａｔｅを信号線５０４を介して取り込む。このＩｎｖａｌｉｄａｔｅを受け取った下位バス判定回路３１４は、上位トランザクションバッファ３１２に格納されているＷｒｉｔｅＢａｃｋ要否の判定結果（前記ステップ(3.3)参照）を読み出し、これに従ってＷｒｉｔｅＢａｃｋ指令信号を生成する。

【０２９９】この動作例では、ステップ(3.3)にてＷｒｉｔｅＢａｃｋが必要と判定されているので、下位バス出力判定回路３１４は、アドレスＢのデータを書き戻すためのＷｒｉｔｅＢａｃｋの発行を指示する指令信号を生成する。この指令信号は、下位バストランザクション生成・保持回路３３０に出力する。

【０３００】(11.2)これを受け取った下位バストランザクション生成・保持回路３３０は、アドレスＢについてのＷｒｉｔｅＢａｃｋトランザクションを生成する。そして、このＷｒｉｔｅＢａｃｋは、下位バスアービトレーション回路３２８により下位バスの使用権が取得でき次第、信号線５１６を経由して下位バスに発行される。

【０３０１】(11.3)また、下位バス出力判定回路３１４からのＷｒｉｔｅＢａｃｋ指示は、キャッシュ制御回路３２２にも入力される。この指示を受けたキャッシュ制御回路３２２は、エビクションバッファ３３６からアドレスＢのデータを出力させる。このデータは、信号線５２２を経由して下位バスに出力される。

【０３０２】このようにして、バスブリッジ＃０からのＷｒｉｔｅＢａｃｋによって、アドレスＢのデータがメインメモリに書き戻される。

【０３０３】実施形態２．以上説明した実施形態１は、階層バスシステムにおいて、バスの電気的な制約を克服して一つのシステム上に接続できるＣＰＵの数を増やすものであった。

【０３０４】ところで、一つの階層バスシステム上に接続可能なＣＰＵの数は、そのような物理的な制約のみならず論理的な制約を受けることもある。

【０３０５】すなわち、同時に複数のバストランザクションのスプリットを許容する場合、各トランザクションに対するスプリット応答を正確に発行元のＣＰＵに返すためにトランザクションにＩＤを持たせる。このＩＤをスプリットＩＤと呼ぶ。従来の階層バスシステムでは、このスプリットＩＤは、図８に示すように次のように定義されていた。

【０３０６】

【数１】

スプリットＩＤ＝ＣＰＵ＿ＩＤ＋トランザクションＩＤ

ここで、ＣＰＵ＿ＩＤは、階層バスシステム内の各ＣＰＵごとに一意的に定められる。また、トランザクションＩＤは、各ＣＰＵ内においてトランザクションごとに一意的に定められる。このようなスプリットＩＤをみれば、バス上のトランザクションが一意的に特定され、またスプリット応答の宛先が特定される。

【０３０７】ところが、このようなスプリットＩＤでは、システム上に接続可能なＣＰＵの個数がＣＰＵ＿ＩＤのビット数によって制限されてしまう。例えば、ＣＰＵ＿ＩＤに２ビットが割り当てられている場合は、システムに接続可能なＣＰＵの個数は４個に限定される。これがＣＰＵ接続個数に対する論理的な制約である。

【０３０８】従って、実施形態１のごとくシステムの階層化を行い、物理的な制約を回避したとしても、従来のようなスプリットＩＤの割り当て方では、上記論理的な制約によりシステムに接続可能なＣＰＵの個数が限られてしまう。

【０３０９】本実施形態２は、このような論理的な制約を回避しようとするものである。この実施形態は、図１に示したバスブリッジを用いた階層型のシステムに適用される。

【０３１０】本実施形態では、上記論理的な制約を回避するために、各バスブリッジにそれぞれ固有のＩＤ（以下、ブリッジＩＤと呼ぶ）を与え、下位バス上でのトランザクションのスプリットＩＤを次のように定義する。

【０３１１】

【数２】下位バススプリットＩＤ＝ブリッジＩＤ＋ＣＰＵ＿ＩＤ＋トランザクションＩＤ

したがって、本実施形態における上位バス及び下位バス上のトランザクションのスプリットＩＤの構成は図９に示すようになる。

【０３１２】ここで、ＣＰＵ、バスブリッジ、Ｉ／Ｏなどの各エージェントのＩＤは、各バスごとに一意的になるように定める。例えば、図１では下位バスに複数のバスブリッジ、メインメモリ、及びＩ／Ｏ装置が接続されているが、それら各エージェントにＩＤを与える場合には、下位バス内において一意的なＩＤを与える。また、各上位バスについても各ＣＰＵ及びバスブリッジに対して上位バス内において一意的なＩＤを与える。このようなＩＤ割り当てを行った場合、バスブリッジに対しては、上位バスについてのＩＤと下位バスについてのＩＤの２種類のＩＤが与えられる。したがって、図９に示す下位バススプリットＩＤにおけるブリッジＩＤは、バスブリッジの２つのＩＤのうち下位バスについてのＩＤを指す。

【０３１３】この方式において、上位バスのトランザクションに対するスプリットＩＤは、その上位バスを発行したＣＰＵのＩＤと、そのＣＰＵにおける当該トランザクションのＩＤから定められる。そして、上位バス・トランザクションをスプリットして下位バスにトランザクションを発行した場合、その下位バス・トランザクションには、元の上位バス・トランザクションのＣＰＵ＿ＩＤ及びトランザクションＩＤに当該スプリットを行ったバスブリッジの下位バスに対するブリッジＩＤを加えたものを下位バススプリットＩＤとして与える。このような下位バススプリットＩＤによれば、各ブリッジから下位バスに発行されたトランザクションを一意的に特定することができる。

【０３１４】下位バススプリットＩＤの割り付けはバスブリッジによって行われる。より具体的に、バスブリッジとして図６に示す構成の装置を用いた場合には、下位バススプリットＩＤの割り付けは、下位バストランザク

ション生成・保持回路３３０で行われる。

【０３１５】このようなスプリットＩＤ割り付け方式によれば、１つの上位バスに接続可能なＣＰＵの個数は従来同様限定されるものの、システム全体では、上位バスの数を増やすことによってＣＰＵの個数を増やすことが可能となる。

【０３１６】変形例１．次に、実施形態２の下位バススプリットＩＤ割り当て方式の変形例１について説明する。

【０３１７】図１０は、この変形例１の方式によるスプリットＩＤの構成を示す説明図である。図に示すように、この変形例では、下位バススプリットＩＤを割り当てるときに、上位バススプリットＩＤに単にブリッジＩＤを付加するのではなく、上位バススプリットＩＤのビット数を縮退したものを下位バスにおけるトランザクションＩＤとする。そして、この下位バス・トランザクションＩＤにブリッジＩＤを付加することにより、下位バススプリットＩＤのビット数を上位バススプリットＩＤのビット数と等しくする。ここで、上位バススプリットＩＤから下位バスにおけるトランザクションＩＤを生成する際には、この下位バス・トランザクションＩＤがスプリットを行ったバスブリッジにおいてトランザクションごとに一意的となるようにする。

【０３１８】このように下位バススプリットＩＤを定めることにより、下位バスに対して上位バス用のデバイス（例えばＣＰＵ）を接続することが可能となる。

【０３１９】なお、この構成においては、上位バスのスプリットＩＤとこれを圧縮したトランザクションＩＤとの対応関係を管理するＩＤ管理手段をバスブリッジに設ける。　変形例２．次に、実施形態２の下位バススプリットＩＤ割り当て方式の変形例２について説明する。

【０３２０】図１１は、この変形例２の方式によるスプリットＩＤの構成を示す説明図である。図に示すように、この変形例では、ブリッジＩＤをＣＰＵ＿ＩＤと同ビット数になるように設定する。そして、下位バススプリットＩＤを割り当てるときに、上記変形例１と同様、上位バススプリットＩＤを変形して下位バスにおけるトランザクションＩＤを生成する。このとき、本変形例では、上位バスでのトランザクションＩＤのビット数と等しくなるように上位バススプリットＩＤを縮退して、下位バス・トランザクションＩＤを生成する。そして、この下位バス・トランザクションＩＤにブリッジＩＤを付加することにより、下位バススプリットＩＤのビット数を上位バススプリットＩＤのビット数と等しくする。

【０３２１】この変形例２の方式によれば、上位バススプリットＩＤと下位バススプリットＩＤとが同ビット数となるだけでなく、上位バススプリットＩＤにおけるＣＰＵ＿ＩＤと下位バススプリットＩＤにおけるブリッジＩＤとが同ビット数となる。したがって、この方式によれば、バスブリッジとＣＰＵとを論理的に等価に扱うことができ、図１に示した下位バスに対してＣＰＵを接続することが可能となり、システム構成の柔軟性が向上する。

【０３２２】なお、以上では、バスブリッジから下位バスに発行するトランザクションに対して与えるスプリットＩＤの生成方式について説明したが、この変形例の方式はバスブリッジから上位バスに発行するトランザクションに与えるスプリットＩＤにも応用することができる。この場合、バスブリッジは、下位バスから受け取ったトランザクションのスプリットＩＤを変形してトランザクションＩＤを生成し、このトランザクションＩＤに対して当該バスブリッジの上位バスにおけるブリッジＩＤを付加することにより上位バススプリットＩＤを生成する。

【０３２３】この変形例２における具体的なスプリットＩＤの割り付け方について説明する。まず、上位バスに発行されたトランザクションをバスブリッジがスプリットして下位バスにトランザクションを発行する場合におけるスプリットＩＤの割り付け方から説明する。

【０３２４】この例では、バスブリッジ内に図１２に示すような管理テーブルを設け、この管理テーブルを用いて下位バススプリットＩＤを決定する。図１２に示す管理テーブルは、トランザクションＩＤの数の分だけの欄から構成される。例えば、トランザクションＩＤが８ビットである場合には、管理テーブルは２５６の欄から構成される。管理テーブルの各欄には、当該トランザクションＩＤが使用中であるか未使用であるかを示す使用／未使用フラグが設けられる。

【０３２５】上位バスのトランザクションをスプリットすると決定した場合、バスブリッジは、図１２の管理テーブルから未使用の欄を探し出し、この結果見付かった未使用欄のトランザクションＩＤを、下位バスに対して発行するトランザクションのトランザクションＩＤとする。次に、このトランザクションＩＤに当該バスブリッジのブリッジＩＤを付加して下位バススプリットＩＤとする。そして、バスブリッジは、このようにして求められた下位バススプリットＩＤをヘッダ等に付加して、下位バスに対してトランザクションを発行する。また、このとき、管理テーブルの当該トランザクションＩＤの欄には、スプリットを行った上位バス・トランザクションのスプリットＩＤが登録され、使用／未使用フラグが「未使用」から「使用中」に変更される。そして、「使用中」となった欄は、スプリットしたトランザクションに対するスプリット応答トランザクションを発行する際に、使用／未使用フラグが「未使用」に変更される。これにより、その欄に対応するトランザクションＩＤは再び使用可能となる。

【０３２６】例えば、バスブリッジとして図６に示す構成を用いた場合、この管理テーブルは上位バストランザクションバッファ３１２に格納される。上位トランザク

ションバッファ３１２は、信号線５０４から入力されるスプリットＩＤと下位バス出力判定回路３１４からのスプリットについての情報に基づいて前述したような管理テーブルのメンテナンスを行う。すなわち、バスブリッジが上位バスからのトランザクションを受け取ったとき、その上位バストランザクションのスプリットＩＤが信号線５０４を経由して上位トランザクションバッファ３１２に入力される。そして、下位バス出力判定回路３１４にてトランザクションのスプリットが決定されると、この決定を示す信号が上位トランザクションバッファ３１２に入力される。すると、上位バストランザクションバッファ３１２は、管理テーブルの未使用欄を探し、見付かった未使用欄に上位バスのスプリットＩＤを格納し、その欄のフラグを「使用中」に変更する。同時に、上位バストランザクションバッファ３１２は、その欄の番号（トランザクションＩＤに対応）を示す信号を下位バストランザクション生成・保持回路３３０に与える。下位バストランザクション生成・保持回路３３０は、受け取った欄番号に対応するトランザクションＩＤを求め、このトランザクションＩＤに当該バスブリッジの下位バス用のブリッジＩＤを付加して下位バススプリットＩＤを生成する。

【０３２７】以上説明したように、図１２に示すような管理テーブルを用いて各トランザクションＩＤの使用／未使用を管理し、上位バストランザクションをスプリットする場合には、それらトランザクションＩＤのうち未使用のものを順に下位バススプリットＩＤ中のトランザクションＩＤとして使用することにより、下位バスに対して規定されているトランザクションＩＤを無駄なく使用することが可能になる。すなわち、例えば下位バスにおけるトランザクションＩＤを、上位バススプリットＩＤから関数などを用いて定める場合、異なった上位バススプリットＩＤが同じ下位バストランザクションＩＤにマッピングされる可能性がある。この場合、後から来た上位バストランザクションはスプリットすることができず、後の上位バストランザクションについてはＣＰＵに対してリトライをさせることになる。これに対して本構成のごとく管理テーブルを用いた場合では、異なった上位バストランザクションに対して同一の下位バストランザクションＩＤが割り当てられることはなく、トランザクションＩＤを無駄なく使用することができる。また、リトライの発生頻度を減らすことができるので、システム全体のスループット、レイテンシを向上させることができる。

【０３２８】なお、この方式においては、リトライが生じるのは管理テーブルに未使用の欄がなくなった場合のみである。このような場合において、上位バスからトランザクションが来た場合は、バスブリッジは上位バスにリトライ終了信号を返し、ＣＰＵに同一トランザクションの再発行を要求する。

【０３２９】次に、下位バスのトランザクションを受け取ったバスブリッジが上位バスにトランザクションを発行する場合の、上位バストランザクションに対するスプリットＩＤの割り付け方について説明する。

【０３３０】この場合は、バスブリッジ内に図１３に示すような管理テーブルを設け、この管理テーブルを用いて上位バススプリットＩＤを決定する。図１３の管理テーブルは、図１２に示したものと同様の構成を有している。

【０３３１】バスブリッジが、下位バスのトランザクションを受けて上位バスに対してトランザクションを発行する場合、前記図１２の場合と同様にして、図１３の管理テーブルの未使用欄のトランザクションＩＤを、上位バスに対して発行するトランザクションのトランザクションＩＤとする。次に、このトランザクションＩＤに当該バスブリッジの上位バスに対するブリッジＩＤを付加して上位バススプリットＩＤとする。このとき、管理テーブルの当該トランザクションＩＤに対応する欄には、下位バスバス・トランザクションのスプリットＩＤが登録され、使用／未使用フラグが「未使用」から「使用中」に変更される。そして、「使用中」の欄は、上位バスに発行したトランザクション終了したときに「未使用」に変更される。これにより、その欄に対応するトランザクションＩＤは再び使用可能となる。

【０３３２】例えば、バスブリッジとして図６に示す構成を用いた場合、この管理テーブルは下位バストランザクションバッファ３２６に格納される。下位バストランザクションバッファ３２６は、信号線５１６から入力される下位バススプリットＩＤと、上位バス出力判定回路３１６から入力される上位バスに対するトランザクション発行の有無についての情報とに基づき、管理テーブルをメンテナンスする。すなわち、上位バス出力判定回路３１４にて上位バスに対してトランザクションを発行すると決定された場合、下位バストランザクションバッファ３２６の管理テーブルの未使用欄に元の下位バストランザクションのスプリットＩＤが登録され、その欄の番号を示す信号が上位バストランザクション生成・保持回路３０４に入力される。そして、上位バストランザクション生成・保持回路３０４は、その欄番号に対応するトランザクションＩＤを求め、このトランザクションＩＤに当該バスブリッジの上位バス用のブリッジＩＤを付加して上位バススプリットＩＤを生成する。

【０３３３】このように、図１３に示す管理テーブルを用いて上位バススプリットＩＤを管理することにより、異なった下位バススプリットＩＤに対して同一の上位バススプリットＩＤが割り付けられることがなくなり、トランザクションＩＤを無駄なく使用することができる。したがって、上位バススプリットＩＤに空きがない場合に起こる下位バストランザクションのウエイト（ｗａｉｔ：上位バスにトランザクションが発行可能となるまで

下位バスのトランザクションを待たせること。この間下位バスは当該トランザクションによって占有される）処理の頻度も減らすことができ、システム全体のスループット、レイテンシを向上させることができる。

【０３３４】なお、この方式において下位バストランザクションのウエイトが生じるのは、管理テーブルに未使用欄がなくなった場合のみである。

【０３３５】実施形態３．実施形態１においては、階層バスシステムにおける上位バスの仕様と下位バスの仕様との関係については特に規定していなかった。

【０３３６】これに対して本実施態は、上位バスの仕様と下位バスの仕様とに一定の関係を規定することにより、階層バスシステムのシステム構成の柔軟性を向上させることを目的とするものである。

【０３３７】すなわち、本実施形態では、バスの物理的仕様及び論理的仕様の両方について、上位バスと下位バスとで仕様の統一を図る。ここで、バスの物理的仕様には、バス上の信号の電気レベルやバスが動作可能なクロック周波数などがある。また、論理的仕様としては、バス上のトランザクションの種類及びそのビットパタン、トランザクションの要求アドレスの範囲及びその表現形式、バスに接続されるキャッシュが出力するスヌープ出力の表現形式及びそのビットパタン、バス上におけるトランザクションの識別子であるスプリットＩＤの構成などがある。なお、スプリットＩＤの構成の仕様の統一には、前記実施形態２の変形例２において説明したスプリットＩＤ構成方式、すなわち上位バススプリットＩＤと下位バススプリットＩＤとのビット幅を同一にし、上位バススプリットＩＤにおけるＣＰＵ＿ＩＤと下位バススプリットＩＤにおけるブリッジＩＤとのビット幅を同一にすることが含まれる。

【０３３８】このように上位バスと下位バスとの仕様を同一にすることにより、以下に示すようにシステム構成のバリエーションを広げることができる。

【０３３９】まず、図１４に示すように、下位バスに対して、上位バス上のＣＰＵ＃０～＃２と同等のＣＰＵ＃３を接続可能となる。このとき、下位バスに接続されたバスブリッジ及びＣＰＵ＃３にはそれぞれ下位バス上で一意的なＩＤを与え、これをスプリットＩＤ生成の際に用いる。具体的なスプリットＩＤの生成方式としては、実施形態２のものを用いることができる。

【０３４０】また、同一バス上にＣＰＵとバスブリッジが接続可能という点を拡張すれば、バス上のＣＰＵをバスブリッジと置き換えることにより、図１５に示すような多階層のシステムの構築が可能となる。この場合、各バスに接続されたエージェント（ＣＰＵ、バスブリッジ）には、当該バス上において一意的なＩＤが与えられ、このＩＤにより当該バス上におけるトランザクションのスプリットＩＤを生成する。具体的なスプリットＩＤの生成方式としては、実施形態２のものを用いることができる。なお、図１５において、第２上位バスのＣＰＵをバスブリッジに置き換えることにより、さらに多階層のシステムを構築することもできる。

【０３４１】また、本実施形態において、実際的なシステム構成の柔軟性を向上させるためには、モジュール化の手法を採用することもできる。

【０３４２】すなわち、コンピュータ本体には、メインメモリ及びＩ／Ｏ装置等を接続した下位バス構成のみを設けておき、ＣＰＵ等の上位の構成はボード化して適宜交換可能とする。交換ユニットたるボードとしては、一つのＣＰＵのみを実装したボードや、１つ以上のＣＰＵとバスブリッジ及びブリッジキャッシュとを１本のバスに接続して構成した１つのクラスタを実装したボード等を用いることができる。これら各種のボードは、下位バスに接続可能なインタフェースを有する。

【０３４３】この方式によれば、例えば図１６に示すように、１つＣＰＵのボード１００を接続した１ＣＰＵのシステムから、クラスタを収容したボード１０２を接続したマルチＣＰＵのシステムに容易にバージョンアップが可能となる。このように、この方式によれば、ユーザ等の要望に合わせてシステム構成を容易に変更することが可能となる。

【０３４４】また、図１７に示すように、ＣＰＵ、メインメモリ及びＩ／Ｏ装置を含む基本的なシステムに対して、追加ボードとしてクラスタを収容したボード１０２を接続するという構成も可能である。

【０３４５】更には、下位バスに対して複数のボードを並列接続することにより、システムの性能を更に向上させることも可能である。

【０３４６】実施形態４．以上説明した各実施形態は、複数のＣＰＵが１つの下位バスを介して１つのメインメモリを共有するシステムについてのものであった。これに対して、本実施形態は、下位バス－メインメモリを複数系統設け、これらメインメモリをそれぞれ上位の複数のＣＰＵで共有する。

【０３４７】図１８は、実施形態４に係る階層バスシステムの一例の構成を示す説明図である。図１８において、図１と同一の部分には同一の番号を付してその説明を省略する。

【０３４８】図１８に示すシステムは、２つの下位バス４０ａ，４０ｂを有し、下位バス４０ａにはメインメモリ５０ａ及びＩ／Ｏ装置５２ａが、また下位バス４０ｂにはメインメモリ５０ｂ及びＩ／Ｏ装置５２ｂが、それぞれ接続されている。そして、このシステムにおいては、各上位バスには、それぞれ２つ下位バスにそれぞれ接続するために、２系統のバスブリッジ及びブリッジキャッシュが設けられている。すなわち、上位バス２０には、下位バス４０ａと接続するためのバスブリッジ３０ａ及びブリッジキャッシュ３１ａと、下位バス４０ｂと接続するためのバスブリッジ３０ｂ及びブリッジキャッ

シュ３１ｂとが接続されている。また、上位バス２２には、下位バス４０ａと接続するためのバスブリッジ３２ａ及びブリッジキャッシュ３３ａと、下位バス４０ｂと接続するためのバスブリッジ３２ｂ及びブリッジキャッシュ３３ｂとが接続されている。

【０３４９】この構成において、メインメモリ５０ａ及びメインメモリ５０ｂにはそれぞれ異なったアドレス空間が割り当てられる。したがって、各バスブリッジは、自身が接続されている下位バスのメインメモリのアドレス空間を自らの受け持ち範囲とする。すなわち、各バスブリッジは、上位バスから受け取ったトランザクションの要求アドレスが自らの受け持ち範囲か否かを判定する機構を有し、受け持ち範囲であると判定された場合にのみ、その上位バス・トランザクションに対する応答処理を行う。例えば、ＣＰＵ１０から発せられたトランザクションがメインメモリ５０ｂのアドレスに対するものであった場合、バスブリッジ３０ａ，３０ｂは共にそのトランザクションを受け取ってアドレスが各自の受け持ち範囲に含まれるか否かを判定する。この結果、バスブリッジ３０ｂがそのトランザクションに対する処理を行うことになる。そして、バスブリッジ３０ｂは、ブリッジキャッシュ３１ｂのスヌープを行い、この結果に基づいて必要に応じてトランザクションのスプリットなどの処理を行う。

【０３５０】したがって、この構成では、上位バス２０、２２に接続された各ＣＰＵ１０～１５は、下位バス及びメインメモリが２系統存在することを意識せずにトランザクションを発行することができる。

【０３５１】このように、本実施形態では、システムに複数のメインメモリを設け、各メインメモリを別々の下位バスに接続したことにより、各下位バスの負荷を低減し、下位バスのスループットを向上させることができる。

【０３５２】なお、上記構成において、各バスブリッジの受け持ち範囲を、コンピュータ起動時のセルフテストにおいて判明した各メインメモリの使用可能アドレス範囲に基づき調節することもできる。この場合、各バスブリッジは、セルフテストで判明した使用可能アドレス範囲情報を得て、各自の接続されている下位バスのメインメモリの使用可能アドレス範囲を自らの受け持ちアドレス範囲として設定する。このような方法によれば、各バスブリッジの受け持ち範囲を最適化することができる。

【０３５３】また、図１８の例では、各下位バス４０ａ，４０ｂにそれぞれＩ／Ｏ装置５２ａ又は５２ｂが接続されていたが、そのような構成では、トランザクションのアドレスによって使用可能なＩ／Ｏ装置が限定される。すなわち、あるトランザクションが、メインメモリ５０ａのアドレスに対するものの場合は、そのトランザクションはＩ／Ｏ装置５２ａのみしか使用できない。逆にいえば、すべてのトランザクションが、アドレスにかかわりなく同一のＩ／Ｏ装置を利用できるようにするためには、複数の下位バスにそれぞれ同一のＩ／Ｏ装置を接続する必要がある。

【０３５４】図１９は、このような問題を解決するためのシステム構成を示す説明図である。図１９において、図１８と同一の部分には同一の符号を付してその説明を省略する。

【０３５５】図１９の構成では、１つのＩ／Ｏ装置５２が２つの下位バス４０ａ，４０ｂによって共有される。すなわち、Ｉ／Ｏ装置５２はＩ／Ｏバス５６に接続され、下位バス４０ａ及び４０ｂはそれぞれバスブリッジ５４ａ及び５４ｂを介してＩ／Ｏバス５６に接続されている。バスブリッジ５４ａ及び５４ｂにはブリッジキャッシュは設けられない。バスブリッジ５４ａ及び５４ｂは、下位バス４０ａ又は４０ｂから受け取ったトランザクションがＩ／Ｏ装置５２に対するトランザクションであると判定した場合は、そのトランザクションをＩ／Ｏバス５６に転送する。そして、Ｉ／Ｏ装置５２は、Ｉ／Ｏバス５６からトランザクションを受け取り、必要な処理を行う。

【０３５６】図１９の構成によれば、各トランザクションは、アドレスにかかわらず、システムが有するすべてのＩ／Ｏ装置を使用することができる。

【０３５７】なお、図１９の構成において、バスブリッジ５４ａ，５４ｂが、下位バス４０ａ又は４０ｂからＩ／Ｏバス５６に対してトランザクションを転送するときに、スプリットを行うようにしてもよい。この場合、Ｉ／Ｏトランザクション処理時における下位バスの負荷を低減することができる。

【０３５８】また、バスブリッジとブリッジキャッシュとを１つのパッケージとして構成し、ブリッジキャッシュを使用するモードと使用しないモードとを切り換え可能とすることにより、一種類のバスブリッジ・パッケージで上位バス－下位バス間用と下位バス－Ｉ／Ｏバス間用の両方に用いることができる。

【図面の簡単な説明】

【図１】　本発明に係る階層バスシステムの全体的な構成の一例を示す概略構成図である。

【図２】　実施形態１における変形例の概略構成を示す説明図である。

【図３】　エビクションバッファを設けたバスブリッジの概略構成を示す説明図である。

【図４】　補助キャッシュを設けたバスブリッジの概略構成を示す説明図である。

【図５】　リトライ判定機能を有するバスブリッジの概略構成を示す説明図である。

【図６】　実施形態１におけるバスブリッジ及びブリッジキャッシュの構成を示すブロック図である。

【図７】　階層バスシステムの一例を示す図である。

【図８】　従来のスプリットＩＤの構成を示す図であ

る。

【図９】　実施形態２における上位バススプリットＩＤと下位バススプリットＩＤの構成を示す図である。

【図１０】　実施形態２の変形例１における上位バススプリットＩＤと下位バススプリットＩＤの構成を示す図である。

【図１１】　実施形態２の変形例２における上位バススプリットＩＤと下位バススプリットＩＤの構成を示す図である。

【図１２】　実施形態２の変形例２における下位バストランザクションＩＤの管理テーブルを示す図である。

【図１３】　実施形態２の変形例２における上位バストランザクションＩＤの管理テーブルを示す図である。

【図１４】　実施形態３における下位バスにＣＰＵを接続した階層バスシステムの構成を示す図である。

【図１５】　実施形態３における多階層の階層バスシステムの構成を示す図である。

【図１６】　実施形態３におけるボード交換を示す説明図である。

【図１７】　実施形態３におけるボード追加を示す説明図である。

【図１８】　実施形態４に係る階層バスシステムの一例の概略構成を示す説明図である。

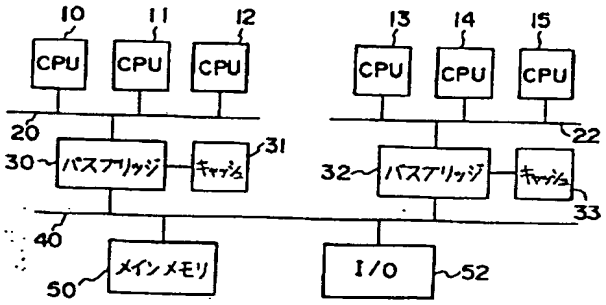
【図１９】　実施形態４において複数の下位バスがＩ／Ｏ装置を共有する構成例を示す説明図である。

【図２０】　バス共有型の密結合マルチプロセッサシステムを示す概略図である。

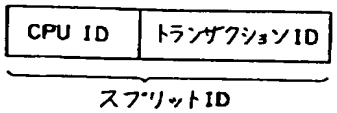
【符号の説明】

１０～１５　ＣＰＵ、２０，２２　上位バス、３０，３２　バスブリッジ、３１，３３　ブリッジキャッシュ、４０　下位バス、５０　メインメモリ、５２Ｉ／Ｏ装置、６０　バスブリッジ制御部、６２　エビクションバッファ、６４アドレスレジスタ、６６　データバッファ、６８　補助キャッシュ、７２　スプリット情報レジスタ、８０　上位キャッシュ状態フラグ記憶部、３００　バスブリッジ、３０２　上位バスアービトレーション回路、３０４　上位バストランザクション生成保持回路、３０６　スプリット応答生成回路、３０８　上位バススヌープ出力生成回路、３１０　レスポンス生成回路、３１２　上位トランザクションバッファ、３１４　下位バス出力判定回路、３１６　上位バス出力判定回路、３１８　タグリード回路、３２０　タグライト回路、３２２　キャッシュ制御回路、３２４　タグ更新回路、３２６　下位トランザクションバッファ、３２８下位バスアービトレーション回路、３３０　下位バストランザクション生成保持回路、３３２　下位バススヌープ出力生成回路、３３４　レスポンス受信回路、３３６　エビクションバッファ、３３８　リトライ判定回路、４０２　タグＲＡＭ、４０４　キャッシュＲＡＭ。

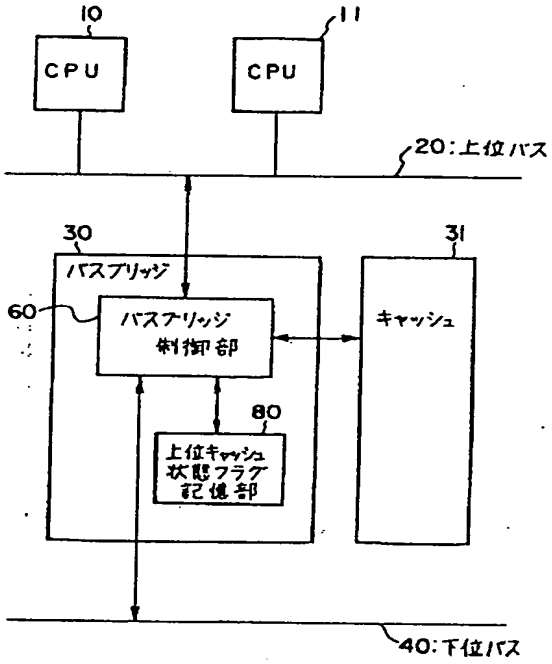
【図１】



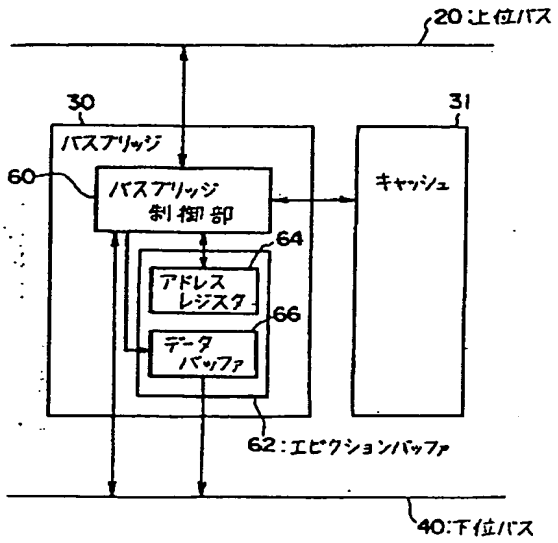
【図８】



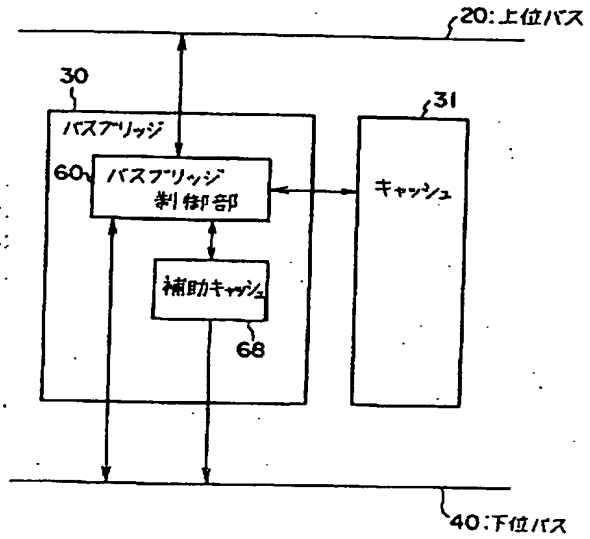
【図２】



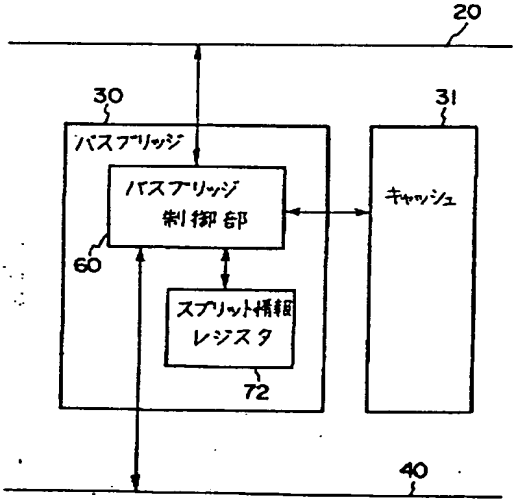
【図３】



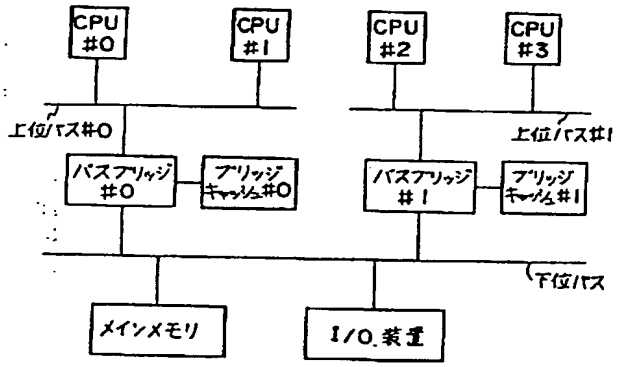
【図４】



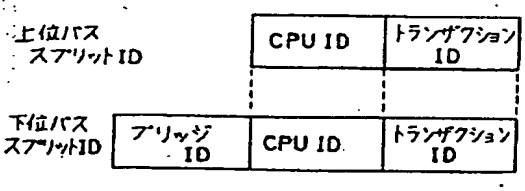
【図５】



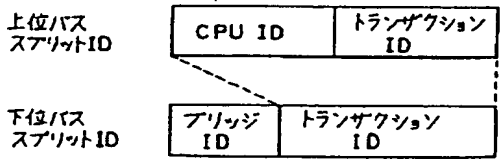
【図７】



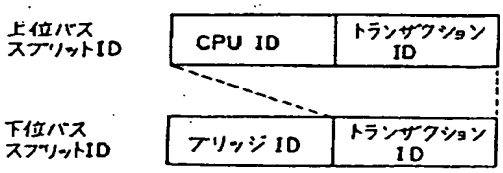
【図９】



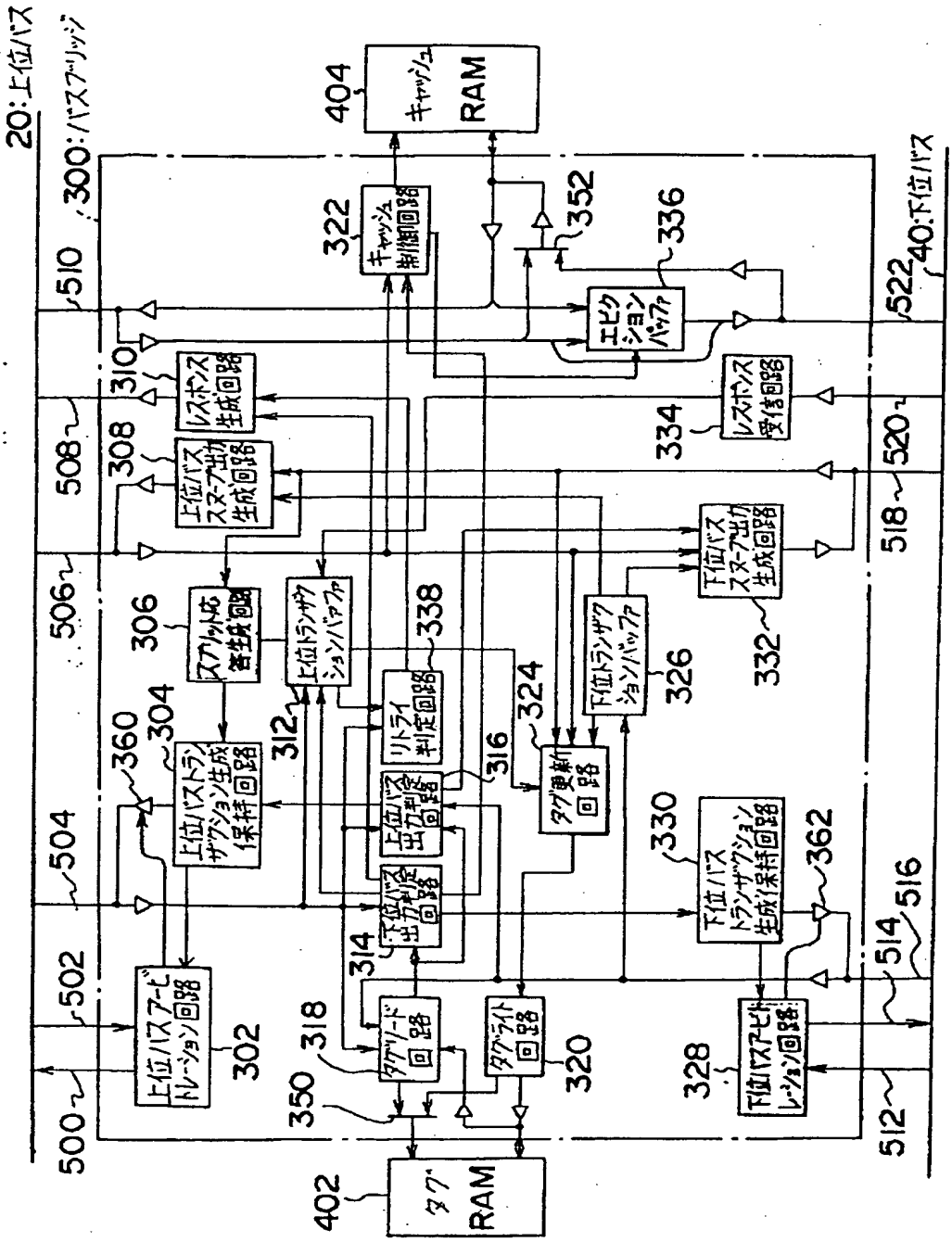
【図１０】



【図１１】



【図6】


20:上位バス
300:バスブリッジ
40:下位バス
404
キャッシュ
RAM
322
キャッシュ制御回路
352
336
エビクションバッファ
522
510
310
レスポンス生成回路
334
レスポンス受信回路
520
508
308
上位バススヌープ出力生成回路
518
506
306
スプリット応答生成回路
上位トランザクションバッファ
338
下位バススヌープ出力生成回路
332
304
360
312
上位バストランザクション生成保持回路
リトライ判定回路
316
324
タグ更新回路
下位トランザクションバッファ
326
504
330
下位バストランザクション生成保持回路
362
314
上位バス出力許可判定回路
下位バス出力許可判定回路
516
514
502
上位バスアービトレーション回路
302
318
350
タグリード回路
タグライト回路
320
328
下位バスアービトレーション回路
500
512
402
タグ
RAM

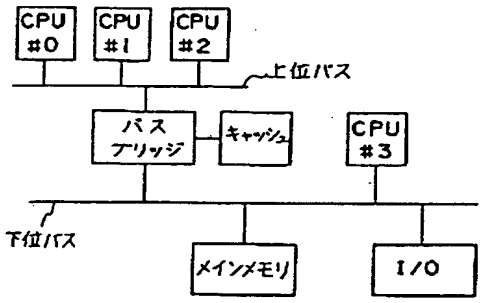
【図１２】

下位バス トランザクションID	上位バス スプリットID	使用/未使用フラグ
0	30	使用中
1	01	使用中
2		未使用
3		未使用

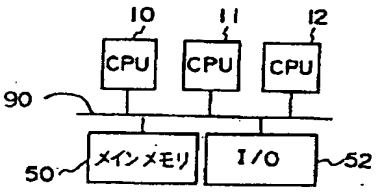
【図１３】

上位バス トランザクションID	下位バス スプリットID	使用/未使用フラグ
0	21	使用中
1	02	使用中
2		未使用
3		未使用

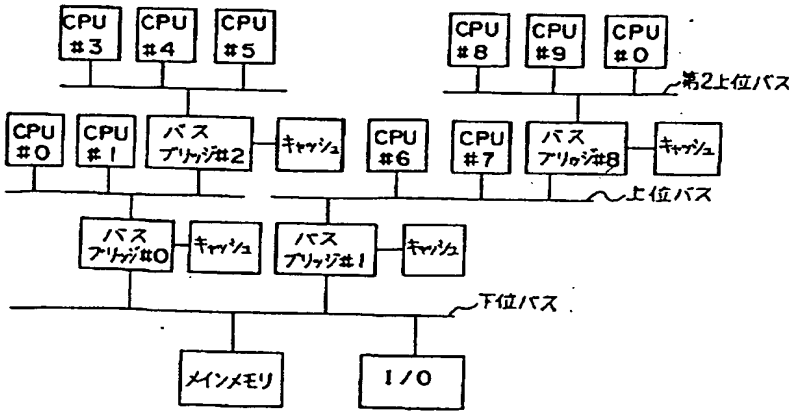
【図１４】



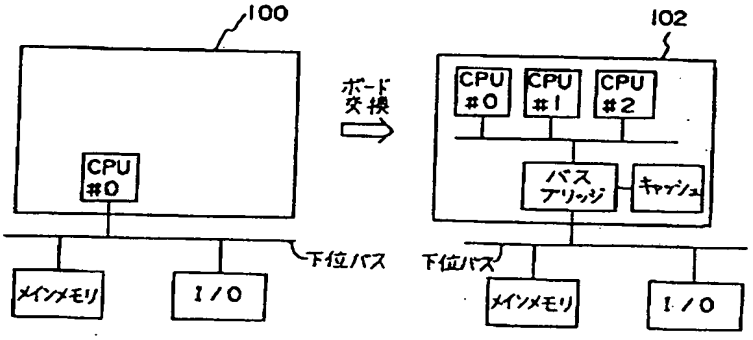
【図２０】



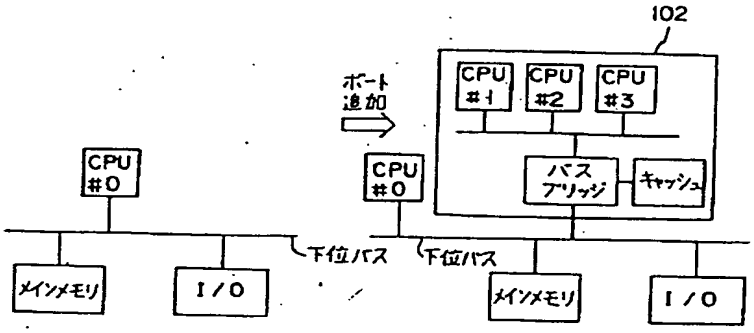
【図１５】



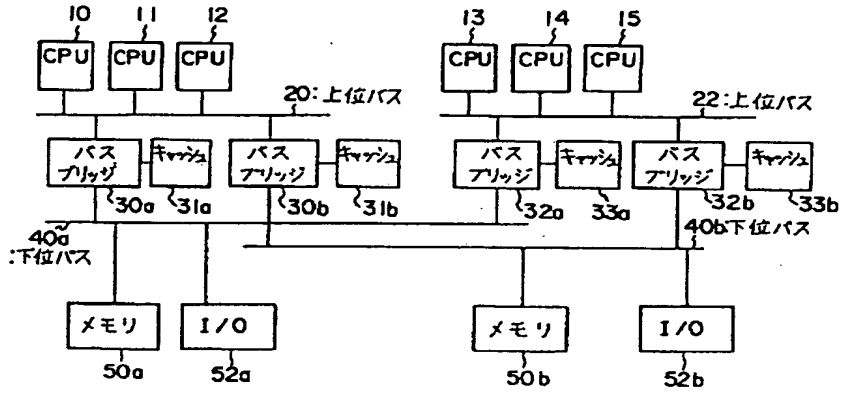
【図１６】


100
102
CPU #0
ボード交換
CPU #0
CPU #1
CPU #2
バスブリッジ
キャッシュ
下位バス
下位バス
メインメモリ
I/O
メインメモリ
I/O

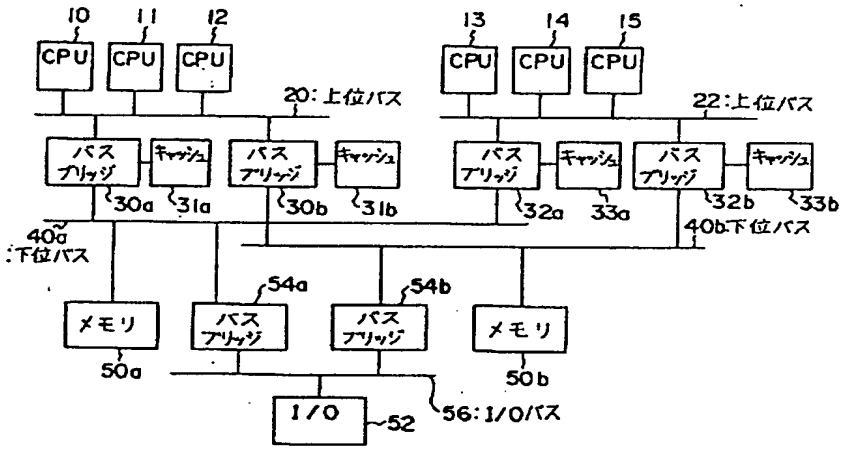
【図１７】


102
ボート追加
CPU #1
CPU #2
CPU #3
CPU #0
CPU #0
バスブリッジ
キャッシュ
下位バス
下位バス
メインメモリ
I/O
メインメモリ
I/O

【図１８】



【図１９】



【手続補正書】
【提出日】平成７年１１月６日
【手続補正１】
【補正対象書類名】明細書
【補正対象項目名】０３１８
【補正方法】変更
【補正内容】
【０３１８】このように下位バススプリットＩＤを定めることにより、下位バスに対して上位バス用のデバイス（例えばＣＰＵ）を接続することが可能となる。なお、この構成においては、上位バスのスプリットＩＤとこれを圧縮したトランザクションＩＤとの対応関係を管理するＩＤ管理手段をバスブリッジに設ける。
【手続補正２】
【補正対象書類名】明細書
【補正対象項目名】０３１９
【補正方法】変更
【補正内容】
【０３１９】変形例２．次に、実施形態２の下位バススプリットＩＤ割り当て方式の変形例２について説明する。